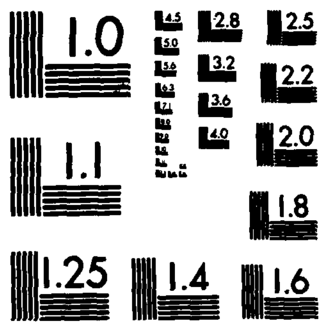
AD-A144 561 ACTIVE SUPPRESSION OF AEROELASTIC INSTABILITIES ON A FORWARD SWEPT WING U..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. G J PASQUINI 1/2

UNCLASSIFIED 01 JUN 84 AFIT/GAE/AA/84J-01 F/G 1/3 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A144 561

AIR FORCE INSTITUTE OF TECHNOLOGY

AIR UNIVERSITY
UNITED STATES AIR FORCE

ACTIVE SUPPRESSION OF AEROELASTIC
INSTABILITIES ON A FORWARD SWEPT WING
USING A LINEAR OPTIMAL REGULATOR

THESIS

AFIT/GAE/AA/84J-01 Glenn J Pasquini

SCHOOL OF ENGINEERING

WRIGHT-PATTERSON AIR FORCE BASE, OHIO

AFIT/GAE/AA/84J-01

ACTIVE SUPPRESSION OF AEROELASTIC

INSTABILITIES ON A FORWARD SWEPT WING

USING A LINEAR OPTIMAL REGULATOR

THESIS

AFIT/GAE/AA/84J-01 Glenn J Pasquini

Approved for public release; distribution unlimited

AFIT/GAE/AA/84J-01

ACTIVE SUPPRESSION OF AEROELASTIC INSTABILITIES ON A FORWARD SWEPT WING USING A LINEAR OPTIMAL REGULATOR

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

Glenn J Pasquini, B.S.

Graduate Aeronautical Engineering

June 1984

Approved for public release; distribution unlimited.

Preface

Present

In recent years, advances in composite materials and active control principles have lead to renewed interest in the Forward Swept Wing (FSW) as a feasible aircraft design alternative. Sweeping a wing forward results in low static divergence speeds and additional aeroelastic instabilities which must be adequately controlled for an aircraft employing this wing design to have an acceptable flight envelope. This thesis seeks to demonstrate ~~Contained herein is a study aimed at demonstrating~~ the utility of applying active feedback control principles to suppress the aeroelastic instabilities associated with an FSW Forward Swept Wing design. Analytical techniques are presented that are useful in the analysis and synthesis of active control laws using optimal control theory methodolgy. cont on p ix

I wish to thank my advisor, Robert A. Calico for his invaluable guidance during the course of this research and my thesis committee members, Peter J. Torvik and Franklin Eastep, for their thorough editing of this document. In addition, I would like to thank my sponsor, Thomas E. Noll, for the use of his research results and for his help in understanding and applying aeroelastic concepts. I would also like to thank Captain Richard Floyd for his assistance in modifying some of the computer programs used to accomplish this project. Finally, I wish to thank my loving wife Linda for her understanding and support during the period of this research.

Contents

List of Figures

List of Tables

List of Symbols

b	reference length, ft
D_i	averaged Pade' denominator coefficients
$h(t)$	wing vertical displacement, ft
J	quadratic cost function
k	reduced frequency
N_i	Pade' numerator polynomial coefficients
s	Laplace variable
$\bar{s}$	modified Laplace variable
$\tilde{s}$	reference area, ft^2
V	freestream velocity, ft/sec
$\delta(t)$	control surface deflection, rad
$\alpha(t)$	wing angle of twist, rad
ω	circular frequency, rad/sec
ρ	freestream density, slug/ft^3
$(\cdots\cdot)$	dot superscripts, indicate time derivatives

List of Symbols (Cont'd.)

Matrices

$[A]$	state coefficient matrix
$[A_i],[a_i]$	aerodynamic coefficient matrices
$[B]$	control coefficient matrix
$[b_i]$	aerodynamic coefficient matrices
$[C]$	output coefficient matrices
$[c_i]$	aerodynamic coefficient matrices
$[G]$	feedback gain matrix
$[K]$	generalized stiffness matrix, observer gain matrix
$[M]$	generalized mass matrix
$[Q]$	state weighting matrix
$[Q(k)]$	general unsteady aerodynamic force matrix
$[P]$	Ricatti solution matrix
$[R]$	control weighting matrix
$[T]$	state transformation matrix
$[\Lambda]$	diagonalized state coefficient matrix

List of Symbols (Cont'd.)

Vectors

$\underline{e}(t)$	reconstruction error
$\underline{F}(t)$	unsteady aerodynamic force vector
$\underline{q}(t), \underline{q}_c(t)$	generalized coordinate vectors
$\underline{u}(t)$	control input vector
$\underline{X}(t)$	state vector
$\hat{\underline{X}}(t)$	state estimate vector
$\underline{Y}(t)$	output vector
$\underline{Z}(t)$	uncoupled state vector
$\hat{\underline{Z}}(t)$	uncoupled state estimate vector
$\underline{\xi}(t)$	observer auxiliary state vector
$\underline{\mu}(t)$	observer control input vector

Abstract

Analytical studies ~~were conducted to~~ investigate the potential of applying optimal control theory techniques to ~~the~~ synthesis of active flutter suppression control laws. For an example application, a ~~Forward-Swept-Wing~~ FSW/Fuselage model previously analyzed by Thomas E. Noll of ~~the Air Force Flight Dynamics Laboratory~~ AFFDL was ~~utilized~~ used. Through ~~the~~ use of Pade' approximants to represent ~~the~~ unsteady aerodynamic forces, the equations of motion are written in standard state space form. Linear optimal regulator theory is then applied to determine particular sets of gains which minimize a quadratic cost function in terms of the states and controls. The control laws are developed at a design flight condition which increases the onset of the lowest instability speed 20% above the wing bending/torsion instability speed. The optimal control law is then applied at off-design flight conditions to assess the robustness of the optimal regulator. Program listings are included.

ACTIVE SUPPRESSION OF AEROELASTIC INSTABILITIES ON A FORWARD SWEPT WING USING A LINEAR OPTIMAL REGULATOR

I. Introduction

In recent years, the forward swept wing has gained renewed interest among the aerospace community as being a feasible aircraft design alternative. This is not a new concept as the forward swept wing has long been recognized as being able to provide improved performance benefits over conventional aft swept wings, provided potential aeroelastic problems could be solved. The aeroelastic instabilities associated with the operation of a forward swept wing include instabilities not typically encountered within the operational flight envelope of conventional aft swept wing aircraft, such as static divergence and body freedom flutter. Conventional passive methods for preventing these instabilities typically result in significant performance penalties to the aircraft, negating the improved performance gained from using a forward swept wing design. Therefore, there is considerable interest in developing other methods of preventing aeroelastic instabilities (or increasing the minimum airspeed at which they occur) that can be used in place of, or in combination with, passive methods.

A method being considered for preventing aeroelastic instabilities involves the utilization of an active feedback control system. An act-

ive control system utilizes an aerodynamic control surface, or surfaces, commanded by signals through an an appropriate control law. The fundamental principles behind the use of this technique have been well documented as a result of the significant amount of research conducted in the 1960s and 1970s to develop active flutter suppression technology (Ref 1-4), and in the early 1980s to advance adaptive control principles. With these advances in active control technology, an active flutter suppression control system is now a feasible solution to the aeroelastic problems associated with forward swept wing aircraft, while offering significantly less weight and performance penalties than alternative solutions.

There have been several studies conducted in recent years (Ref 5-6) pertaining to the use of classical control theory methods for active flutter suppression on aircraft employing a forward swept wing design. Griffen and Eastep initially investigated the use of a simple active feedback control system to control divergence and bending/torsion flutter separately on a cantilever wing configuration. At the time of this study, static divergence was believed to be the most critical aeroelastic instability but when a wing/fuselage model was tested free in pitch (Ref 7-8), body freedom flutter was discovered to occur at a velocity lower than the static divergence speed. This instability is a result of coupling between the airplane rigid body pitch mode (short period) and the wing first bending mode. A more recent study conducted by Noll et al (Ref 9) was directed at developing control laws for a cantilever flexible wing and for a wing/fuselage model free in pitch. In this study, Noll developed a multiple input-multiple output (MIMO) control law by treat-

ing the system as two single input-single output (SISO) sub-systems, and applying classical control theory methodology. Two active control surfaces were utilized along with appropriate feedback compensation, thereby making the control law synthesis an iterative procedure. While his final control law design did accomplish the objectives of the study, it will be demonstrated herein that rather straight forward, optimal control theory techniques can be utilized for the control law synthesis, thereby eliminating the iterative design process. Since in actuality most aircraft control problems are MIMO systems, optimal control theory techniques would seem rather well suited for active control law synthesis.

To apply optimal regulator theory for the synthesis of an active flutter suppression control law, it is first necessary to obtain the aeroelastic equations of motion in the form of constant coefficient differential equations. The problem arises in representing the unsteady aerodynamics for arbitrary motion in constant coefficient differential equation form so as to be incorporated into the equations of motion easily. In the study conducted by Noll, the generalized aerodynamic forces for oscilliatory motion were computed using the doublet-lattice method of Ref 10. To compute the unsteady aerodynamics for arbitrary motion, each generalized force was represented by a Pade' approximant, that is, a ratio of polynomials in the Laplace variable s. This enabled the unsteady aerodynamic forces to be easily incorporated into the transformed equations of motion to be used as a foundation for the control law synthesis procedure.

The purpose of this thesis is to present some analytical techniques that are useful for the analysis and synthesis of active flutter suppres-

sion control laws using optimal regulator theory. Control laws were synthesized for Noll's cantilever forward swept wing model at a design flight condition which increases the onset of the lowest instability speed 20% above the wing bending/torsion instability speed. The optimal control law is then applied at off-design flight conditions to assess the robustness of the optimal regulator. The effect of varying the state weighting matrix at the design flight condition is also assessed along with the adequacy of developing the control law based on a reduced order system model which neglects the high frequency aerodynamic lag states.

II. Theoretical Developement

Selected Model Configuration

It is envisioned that a fighter-type aircraft employing a forward swept wing design could evolve to the point of operational deployment, whereas it would be expected to exhibit operational characteristics similar to those of conventional aft swept wing fighters. Typically, a fighter aircraft exhibiting a multi-role capability will carry externally mounted stores both on the fuselage and under the wings. However, the adverse mass and inertia distribution on the wings caused by the external stores have traditionally resulted in bending/torsion flutter problems occurring within the operational flight envelope. For these reasons, the model configuration chosen by Noll was characterized as having a clamped divergence instability in close proximity to a classical bending/torsion flutter mode, typical of what would occur for a critical external store configuration. Figure 1 contains a schematic of the model selected for the current analyses and shows the key details and dimensions, and the relative sizes of the components.

To calculate the natural frequencies and mode shapes of the forward swept wing configuration, Noll developed a finite element representation of the model for use with the NASTRAN program. For the cantilever analyses, the bar representing the fuselage was restrained in such a manner that no motion was permitted along the bar or wing root. The elastic modes used in the analyses of this study included the first two bending modes and the first torsion mode of the wing. It was determined that

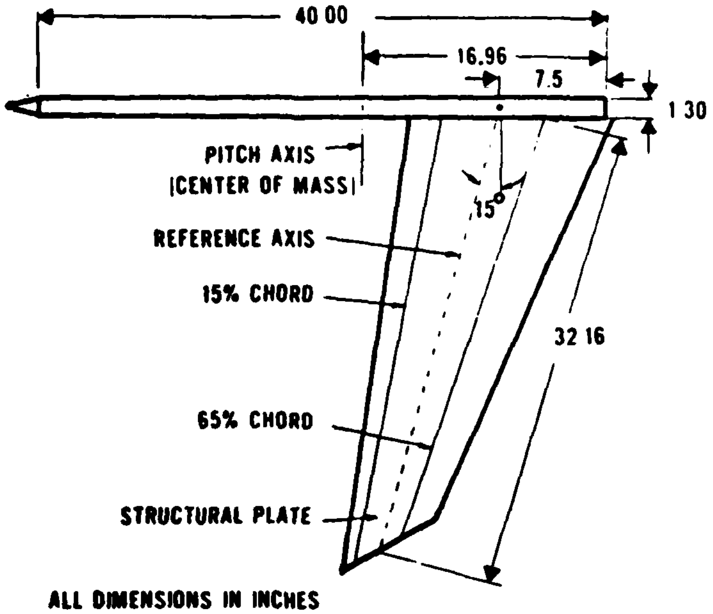


Figure 1. Planform of Forward Swept Wing Model

higher frequency modes could be eliminated when flutter analyses showed that they had no appreciable affect on the instabilities of interest.

State Space Equations of Motion

To apply optimal control theory techniques for control law synthesis, it is desired to obtain the aeroelastic equations of motion as a set of linear first order differential equations of the form

$$\dot{\underline{X}}(t) = [A]\underline{X}(t) + [B]\underline{u}(t) \tag{2-1}$$

where $\underline{X}(t)$ is a real n-dimensional column vector containing the "states" of the system, and $\underline{u}(t)$ is a real m-dimensional column vector which is the input control vector. The equations of motion of a flexible aircraft with control surfaces present can be represented as

$$[M]\ddot{\underline{q}}(t) + [C]\dot{\underline{q}}(t) + [K]\underline{q}(t) + 1/2\rho V^2\tilde{s}[Q(k)]\underline{q}(t) + [M_c]\ddot{\underline{q}}_c(t) + 1/2\rho V^2\tilde{s}[Q_c(k)]\underline{q}_c(t) = 0 \tag{2-2}$$

where the generalized unsteady aerodynamic forces are contained in the $[Q(k)]$ and $[Q_c(k)]$ matrices for the wing and control surfaces respectively. A more detailed derivation of the equations of motion and an explanation of the aerodynamic modeling used in the current analysis is contained in Appendix A and further detail can also be obtained from Ref 9. When the Laplace Transform of Eq 2-2 is taken, substitutions for the unsteady aerodynamics are made and powers of s are equated with time derivatives, resulting in the equations of motion being represented as:

$$\ddot{\ddot{\underline{q}}}(t) + [a_3]\dddot{\underline{q}}(t) + [a_2]\ddot{\underline{q}}(t) + [a_1]\dot{\underline{q}}(t) + [a_0]\underline{q}(t) = [b_3]\dddot{\underline{q}}_c(t) + [b_2]\ddot{\underline{q}}_c(t) + [b_1]\dot{\underline{q}}_c(t) + [b_0]\underline{q}_c(t) \tag{2-3}$$

With the equations of motion expressed in this form, a state space representation can easily be attained (Ref 15) and therefore the system can be modeled as

$$\dot{\underline{X}}(t) = [A]\underline{X}(t) + [B]\underline{u}(t) \tag{2-4}$$

$$\underline{Y}(t) = [C]\underline{X}(t) \tag{2-5}$$

where the state matrices, state vector and control vector are defined as in Appendix A.

Optimal Regulator Theory

With the aeroelastic equations of motion modeled as in Eq 2-4, the control law synthesis procedure can be treated as a linear optimal regulator problem. The function to be minimized is an infinite time integral, quadratic cost function in terms of the states and controls:

$$J = 1/2 \int_0^\infty [\underline{X}^T(t)[Q]\underline{X}(t) + \underline{u}^T(t)[R]\underline{u}(t)]dt \tag{2-6}$$

where [Q] and [R] are weighting matrices on the states and controls respectively. [Q] is chosen to be positive semi-definite and [R] chosen to be positive definite, which guarantees a stable feedback control law. The minimization of this performance index leads to the optimal control law

$$\underline{u}^*(t) = [G]\underline{X}(t) \tag{2-7}$$

where

$$[G] = -[R]^{-1}\ [B]^T\ [P] \tag{2-8}$$

For the time-invariant case (constant coefficient differential equations), [P] is the steady state solution to the Matrix Ricatti equation:

$$-[\dot{P}] = [P][A] + [A]^T[P] - [P][B][R]^{-1}[B]^T[P] + [Q] = [0] \qquad (2\text{-}9)$$

The application of quadratic optimization is an iterative procedure of selecting the appropriate performance function through changes in the weighting matrices [Q] and [R]. The choice of the appropiate weighting matrices is dependent on the designer's past experience and his understanding of the physics of the problem.

Control Law Synthesis

Controller Design. To begin the optimal control law synthesis procedure, it was desired to uncouple the states of the system by utilizing a state transformation to diagonalize the state coefficient matrix [A]. Employing the substitution,

$$\underline{X}(t) = [T]\underline{Z}(t) \qquad (2\text{-}10)$$

where the columns of [T] are the eigenvectors of [A] (Ref 16), the original system represented by Eqs 2-4 and 2-5 can be transformed to:

$$[T]\dot{\underline{Z}}(t) = [A][T]\underline{Z}(t) + [B]\underline{u}(t)$$
$$\underline{Y}(t) = [C][T]\underline{Z}(t)$$

or

$$\underline{\dot{Z}}(t) = [\Lambda]\underline{Z}(t) + [B']\underline{u}(t)$$
$$\underline{Y}(t) = [C']\underline{Z}(t) \qquad (2\text{-}11)$$

where

$$[\Lambda] = [T]^{-1}[A][T]$$
$$[B'] = [T]^{-1}[B]$$

and

$$[C'] = [C][T]$$

With the system modeled in this form, an author-modified version of the program OPTCON (Ref 20) was employed to solve the matrix Ricatti equation given by Eq 2-9, and obtain the optimal control law $\underline{u}^*(t) = [G]\underline{X}(t)$. Due to the large range of numerical values spanned when the controllability matrix is formed inside OPTCON, the Ricatti solver subroutine could not function properly. This resulted in the program defining the system as being uncontrollable when this was clearly not the case.

To eliminate the inherent numerical difficulties, it was decided that the three very high frequency aerodynamic lag states would be neglected for the feedback gain matrix calculation. In general, this would not be necessary if a more efficient Ricatti equation solver could be utilized. The system represented by Eq 2-11 can now be presented as:

$$\begin{Bmatrix} \underline{\dot{Z}}_1(t) \\ \underline{\dot{Z}}_2(t) \end{Bmatrix} = \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix} \begin{Bmatrix} \underline{Z}_1(t) \\ \underline{Z}_2(t) \end{Bmatrix} + \begin{Bmatrix} B_1' \\ B_2' \end{Bmatrix} \underline{u}(t) \qquad (2\text{-}12)$$

where Λ_2 is a 3x3 matrix containing the high frequency roots. The control law was then synthesized based on the following reduced order subsystem of Eq 2-12:

$$\dot{\underline{Z}}_2(t) = [\Lambda_2]\underline{Z}_2(t) + [B_2']\underline{u}(t) \tag{2-13}$$

The above $[\Lambda_2]$ and $[B_2']$ matrices were then input into the modified OPTCON program and the optimal control law

$$\underline{u}'(t)^* = [G_2]\underline{Z}_2(t) \tag{2-14}$$

was determined where $[G_2]$ is formed as in Eq 2-8. This control law was then incorporated into Eq 2-11 by letting:

$$\underline{u}^*(t) = [G_1]\underline{Z}(t) \tag{2-15}$$

where

$$[G_1] = [\ 0 : G_2\ ] \tag{2-16}$$

Now the closed loop system takes on the form:

$$\begin{aligned} \dot{\underline{Z}}(t) &= [\Lambda]\underline{Z}(t) + [B']\underline{u}^*(t) \\ &= [\Lambda]\underline{Z}(t) + [B'][G_1]\underline{Z}(t) \\ &= ([\Lambda] + [B'][G_1])\underline{Z}(t) \\ &= [A'_{CL}]\underline{Z}(t) \end{aligned} \tag{2-17}$$

The closed loop eigenvalues are then determined by the matrix $[A'_{CL}]$.

The optimal control law given by Eq 2-15 can also be put back into the original system by using the inverse relationship:

$$\underline{Z}(t) = [T]^{-1}\underline{X}(t) \tag{2-18}$$

resulting in:

$$\begin{aligned} \underline{u}^*(t) &= [G_1][T]^{-1}\underline{X}(t) \\ &= [G]\underline{X}(t) \end{aligned} \tag{2-19}$$

Therefore,

$$\begin{aligned} \dot{\underline{X}}(t) &= [A]\underline{X}(t) + [B]\underline{u}^*(t) \\ &= [A]\underline{X}(t) + [B][G]\underline{X}(t) \\ &= ([A] + [B][G])\underline{X}(t) \\ &= [A_{CL}]\underline{X}(t) \end{aligned} \tag{2-20}$$

and the resulting eigenvalues of $[A'_{CL}]$ and $[A_{CL}]$ are nearly identical. The coupling of the states resulting from the inverse [T] transformation along with the numerical accuracy of the operations involved does cause some original system closed loop eigenvalues to shift slightly.

The optimal control law given by Eq 2-15 or Eq 2-19 assumes that it is possible to feed back the complete state vector. In actuality, this is not possible as the number of states that are actually measurable are limited by the number of sensors used and how sophisticated these sensors

are. For this study, two independent sensors were utilized; one to sense wing vertical displacement and one to sense wing tip angular acceleration. The results of Noll's analyses indicated that a leading edge surface commanded by displacement feedback provided a reasonable control system design for preventing divergence of the cantilever wing or the body freedom flutter instability associated with the model free in pitch. The displacement sensor was positioned near the intersection of the wing second bending node line and the wing torsion node line. This location was determined to be optimum for feeding back the bending motion of the first elastic mode with minimum inputs from all other elastic modes. Analyses also indicated that a trailing edge control surface commanded by angular acceleration of the wing tip perpendicular to the elastic axis provided an adequate input for controlling the bending/torsion flutter mode. Feeding back wing tip acceleration assured maximum input from the torsion mode with minimum response from the bending modes.

Since the full state vector is not available through direct measurement, it is necessary to design a deterministic observer (state estimator) to reconstruct the state vector from the limited amount of sensor information available.

Observer Design. In order to reconstruct the state $\underline{X}(t)$ of the original system from the observed output vector $\underline{Y}(t)$ as given by Eq 2-5, a linear differential system must be formed whose output is to be an approximation to the state $\underline{X}(t)$. It will be illustrated in the following developement as to what structure this system should have and how it should

behave.

The n-dimensional system represented by

$$\dot{\hat{\underline{X}}}(t) = [F]\hat{\underline{X}}(t) + [G]\underline{Y}(t) + [H]\underline{u}(t) \tag{2-21}$$

where

$$[F] = [A] - [K][C]$$

$$[G] = [K]$$

$$[H] = [B] \tag{2-22}$$

is a full-order observer for the n-dimensional system given in Eq 2-4 if

$$\hat{\underline{X}}(t_0) = \underline{X}(t_0)$$

implies

$$\hat{\underline{X}}(t) = \underline{X}(t) \qquad t \geq t_0 \ ,$$

for all $\underline{u}(t)$, $t \geq t_0$ (Ref 11).

The observer given by Eq 2-21 is called a full-order observer since its state $\hat{\underline{X}}(t)$ has the same dimension as the state $\underline{X}(t)$ of the original system. With the system matrices defined as in Eq 2-22, the observer structure becomes:

$$\dot{\hat{\underline{X}}}(t) = [A]\hat{\underline{X}}(t) + [B]\underline{u}(t) + [K](\underline{Y}(t) - [C]\hat{\underline{X}}(t))$$

or

$$\dot{\hat{\underline{X}}}(t) = ([A] - [K][C])\hat{\underline{X}}(t) + [B]\underline{u}(t) + [K]\underline{Y}(t) \qquad (2\text{-}23)$$

This shows that the stability of the observer is determined by the behavior of the ([A] - [K][C]) matrix. Now, if we consider the observer which is to be designed for the original system, then the reconstruction error

$$\underline{e}(t) = \underline{X}(t) - \hat{\underline{X}}(t) \qquad (2\text{-}24)$$

satisfies the differential equation

$$\dot{\underline{e}}(t) = ([A] - [K][C])\underline{e}(t) \qquad (2\text{-}25)$$

formed by subtracting Eq 2-23 from Eq 2-4. The reconstruction error has the property that

$$\underline{e}(t) \longrightarrow 0 \text{ as } t \longrightarrow \infty$$

for all $\underline{e}(t_0)$, if and only if, the observer is asymptotically stable. Comparing Eq 2-23 and Eq 2-25, we see that the stability of the observer and the asymptotic behavior of the reconstruction error are both determined by the behavior of the matrix ([A] - [K][C]). This clearly shows that the reconstruction error $\underline{e}(t)$ approaches zero, independent of its initial value, if and only if the observer is asymptotically stable. This is a very desirable characteristic of a properly designed observer.

The preceding developement clearly illustrates that deterministic

observer design revolves about determining the gain matrix [K] such that the reconstruction error differential equation is asymptotically stable. Since in this analysis we are dealing with time-invariant coefficient matrices, the stability of the observer is determined by the location of the eigenvalues of the matrix ([A] - [K][C]). We refer to these eigenvalues as the <u>observer poles</u>. It can also be proven (Ref 11) that all observer poles can be arbitrarily located in the complex plane by a suitable choice for [K] if the system is completely reconstructable. In general, it is desirable to choose [K] such that the observer poles are deep in the left-half complex plane so as to obtain fast convergence of the reconstruction error to zero. This generally requires the gain matrix [K] to be large which could cause the observer to be very sensitive to observation noise that may be present.

With the reconstruction error defined as in Eq 2-24, the optimal control law based on the estimate of $\underline{X}(t)$ can be represented as:

$$
\begin{aligned}
\underline{u}^*(t) &= [G]\hat{\underline{X}}(t) \\
&= [G](\underline{X}(t) - \underline{e}(t)) \qquad (2\text{-}26)
\end{aligned}
$$

When this control law is incorporated back into the original system, the following state equation is obtained:

$$
\begin{aligned}
\dot{\underline{X}}(t) &= [A]\underline{X}(t) + [B]\underline{u}^*(t) \\
&= [A]\underline{X}(t) + [B][G](\underline{X}(t) - \underline{e}(t)) \\
&= ([A] + [B][G])\underline{X}(t) - [B][G]\underline{e}(t) \\
&= [A_{CL}]\underline{X}(t) - [B][G]\underline{e}(t) \qquad (2\text{-}27)
\end{aligned}
$$

As previously discussed, for a properly designed observer the error $\underline{e}(t)$ approaches zero over time and the original state system would be attained.

As with the controller, the observer design can be based on the uncoupled $\underline{Z}(t)$ system for computational ease. A vector $\hat{\underline{Z}}(t)$ is defined to be an estimate of $\underline{Z}(t)$ such that

$$\dot{\hat{\underline{Z}}}(t) = [\Lambda]\hat{\underline{Z}}(t) + [B']\underline{u}(t) + [K_1](\underline{Y}(t) - \hat{\underline{Y}}(t))$$
$$\hat{\underline{Y}}(t) = [C']\hat{\underline{Z}}(t) \tag{2-28}$$

A new reconstruction error for the diagonalized system is defined to be:

$$\underline{e}'(t) = \underline{Z}(t) - \hat{\underline{Z}}(t) \tag{2-29}$$

Then by taking the difference between Eq 2-11 and Eq 2-28 the resulting state model for the reconstruction error can be represented as

$$\begin{aligned}\dot{\underline{e}}'(t) &= [\Lambda]\underline{e}'(t) - [K_1][C']\underline{e}'(t) \\ &= ([\Lambda] - [K_1][C'])\underline{e}'(t)\end{aligned} \tag{2-30}$$

and the stability of the error is determined by the eigenvalues of the $([\Lambda] - [K_1][C'])$ matrix. Since the objective is to determine $[K_1]$ such that the reconstruction error is asymptotically stable, existing software can be utilized by forming an auxiliary problem using the duality relationship of Eq 2-28:

$$\dot{\underline{\xi}}'(t) = [\Lambda]^T\underline{\xi}'(t) + [C']\,\underline{\mu}(t) \tag{2-31}$$

The control can be determined using the techniques previously discussed to obtain the desired eigenvalues of the reconstruction error, and hence the observer. Therefore, $\underline{\mu}(t)$ can be represented as

$$\underline{\mu}^*(t) = -[K_1]^T \underline{\xi}'(t) \tag{2-32}$$

and the eigenvalues of the closed loop reconstruction error system are determined by the $([\Lambda] - [C']^T[K_1]^T)$ matrix.

Once again, to prevent numerical difficulties inside the modified OPTCON program, it was necessary to obtain the observer control law using a reduced order model and the same methodology utilized in the controller design. Therefore, the following reduced order auxiliary problem is formed:

$$\dot{\underline{\xi}}_2'(t) = [\Lambda]\underline{\xi}_2'(t) + [C_2']^T\underline{\mu}(t) \tag{2-33}$$

and the control law is given by

$$\underline{\mu}^*(t) = -[K_2]^T\underline{\xi}_2(t) \tag{2-34}$$

The gain matrix $[K_1]$ utilized in Eqs 2-30 and 2-32 to determine the observer poles can be formed from the $[K_2]$ gain matrix in a similar manner that the controller control law was formed in Eq 2-16:

$$[K_1]^T = [\ 0 : K_2^T\ ]$$

or

$$[K_1] = \begin{bmatrix} 0 \\ K_2 \end{bmatrix} \tag{2-35}$$

Once this gain matrix is determined, the reconstruction error can be expressed in terms of the original non-diagonalized system by using the relationship given in Eq 2-10, that is

$$\underline{Z}(t) = [T]^{-1}\underline{X}(t) \quad \text{and} \quad \hat{\underline{Z}}(t) = [T]^{-1}\hat{\underline{X}}(t) \tag{2-36}$$

Now,

$$\begin{aligned} \underline{e}'(t) &= \underline{Z}(t) - \hat{\underline{Z}}(t) \\ &= [T]^{-1}(\underline{X}(t) - \hat{\underline{X}}(t)) \\ &= [T]^{-1}\underline{e}(t) \end{aligned} \tag{2-37}$$

Therefore, Eq 2-30 can be rewritten as:

$$\begin{aligned} \dot{\underline{e}}'(t) &= ([\Lambda] - [K_1][C'])\underline{e}'(t) \\ [T]^{-1}\dot{\underline{e}}(t) &= ([\Lambda] - [K_1][C'])[T]^{-1}\underline{e}(t) \\ \dot{\underline{e}}(t) &= [T]([\Lambda] - [K_1][C'])[T]^{-1}\underline{e}(t) \\ &= [A_e]\underline{e}(t) \end{aligned} \tag{2-38}$$

Finally, an augmented state vector for the entire plant-observer system can be formed using Eqs 2-27 and 2-38:

$$\dot{\tilde{\underline{Z}}}_{CL}(t) = \begin{Bmatrix} \dot{\underline{X}}(t) \\ \dot{\underline{e}}(t) \end{Bmatrix} = \begin{bmatrix} A_{CL} & -BG \\ 0 & A_e \end{bmatrix} \begin{Bmatrix} \underline{X}(t) \\ \underline{e}(t) \end{Bmatrix} \tag{2-39}$$

This augmented state model clearly illustrates that the closed loop system poles consist of two distinct sets; the poles of ([A] + [B][G]) which determine the behavior of the state $\underline{X}(t)$, and the poles of ([A] - [K][C]) which determine the behavior of the error $\underline{e}(t)$. This is known as the Separation Theorem (Ref 14:544) and illustrates that the determination of a feedback control law and the construction of an observer can be considered as separate problems.

III. Results

As outlined in Appendix A, the elements of the state coefficient matrices are functions of velocity and density, and therefore for a fixed altitude the characteristic roots can be determined as velocity is increased from zero. The velocity at which the real part of one of the roots becomes zero is the divergence or flutter velocity depending on whether the frequency is zero or nonzero when it crosses the imaginary axis. Presented in Figure 2 is a velocity root locus for the open loop, unaugmented system at sea level. It was decided to present this plot as a function of equivalent velocity so that a similar plot can be made for any desired altitude by multiplying the velocity by the square root of the density ratio. It should be mentioned here that although these plots are called "root locus plots," they do not adhere to traditional root locus construction rules and just serve to illustrate the movement of the characteristic roots as velocity is varied. The passive divergence speed (V_D) was predicted to occur at 115 fps and the passive bending/torsion flutter speed (V_f) was predicted to occur at 156 fps at a flutter frequency (ω_f) of 104.9 rad/sec. For the augmented analyses, the goal was to determine an optimal control law for the design flight condition which increased the onset of instability 20% above the unaugmented wing bending/torsion flutter speed. To accomplish this goal, the state coefficient matrices were calculated at the design velocity of $1.2V_f$ = 187 fps and utilized in the control law synthesis. With the state model formed, the open loop eigenvalues at this design condition were calculated and are

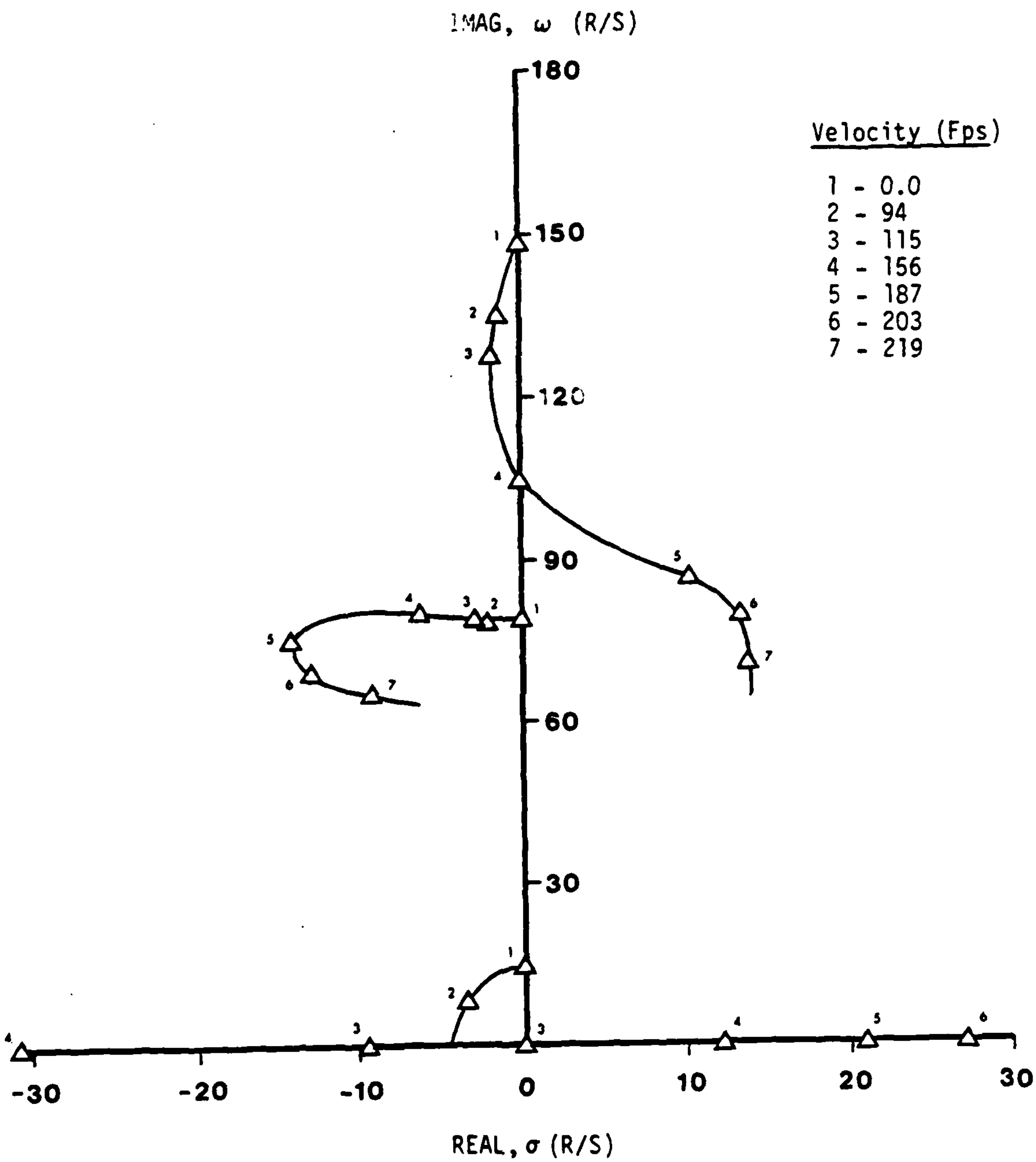


Figure 2. Open Loop Velocity Root Locus
(Altitude = Sea Level)

contained in Table 1.

Table 1

Open Loop Eigenvalues at Design Velocity

No	Real	Imaginary
1	21.190	0.0
2	-57.342	0.0
3	-14.034	74.807
4	-14.034	-74.807
5	10.393	87.113
6	10.393	-87.113
7	-154.03	0.0
8	-178.77	0.0
9	-179.75	0.0
10	-76714.	0.0
11	-76969.	0.0
12	-78118.	0.0

Throughout the control law synthesis procedure, the control weighting matrix [R] was set equal to the identity matrix and only the state weighting matrix [Q] was varied to obtain the desired closed loop behavior. This is an acceptable approach as it is the relative relationship between [Q] and [R] that is of importance. Initially, the state weighting matrix [Q] was set equal to zero since this yields a set of gains that are "cheapest" in terms of input amplitude (Ref 11: 289). Table 2 contains the closed loop eigenvalues at the design condition for [Q] equal to zero. It can be seen that the effect of this control law is to leave all stable eigenvalues unchanged and reflect the unstable eigenvalues about the imaginary axis. It should also be noted that this control law assumes that the full state vector is available for feedback.

Table 2

Closed Loop Eigenvalues Assuming Full State Feedback Gains at Design Velocity

No	Real	Imaginary
1	-21.190	0.0
2	-57.342	0.0
3	-14.034	74.807
4	-14.034	-74.807
5	-10.393	87.113
6	-10.393	-87.113
7	-154.03	0.0
8	-178.77	0.0
9	-179.75	0.0
10	-76714.	0.0
11	-76969.	0.0
12	-78118.	0.0

Table 3

Closed Loop Eigenvalues at Off-design Points

	Eigenvalues	
Velocity(fps)	Real	Imaginary
94 (.6V_f)	-2.843	12.834
	-2.843	-12.834
	-2.283	78.904
	-2.283	-78.904
	-1.290	135.550
	-1.290	-135.550
	-85.303	0.0
	-90.023	0.0
	-90.381	0.0
	-38562.	0.0
	-38690.	0.0
	-39268.	0.0
115 (V_D)	-279.79	0.0
	-9.607	0.0

Table 3 (Cont'd.)

Velocity(fps)	Eigenvalues Real	Imaginary
115 (cont'd.)	-3.026	79.076
	-3.026	-79.076
	-1.707	127.740
	-1.707	-127.740
	-102.58	0.0
	-110.06	0.0
	-110.56	0.0
	-47177.	0.0
	-47333.	0.0
	-48040.	0.0
156 (V_f)	-31.002	0.0
	-30.813	0.0
	-6.296	80.479
	-6.296	-80.479
	-0.200	108.330
	-0.200	-108.330
	-133.63	0.0
	-149.19	0.0
	-149.96	0.0
	-63997.	0.0
	-64212.	0.0
	-65167.	0.0
203 ($1.3V_f$)	-79.674	0.0
	-162.99	0.0
	10.503	72.841
	10.503	-72.841
	-13.000	68.953
	-13.000	-68.953
	-194.05	0.0
	-195.13	0.0
	-392.86	0.0
	-83280.	0.0
	-83556.	0.0
	-84819.	0.0

Table 3 (Cont'd.)

Velocity(fps)	Eigenvalues Real	Imaginary
219 ($1.4V_f$)	-106.17	0.0
	-106.58	0.0
	-9.270	65.287
	-9.270	-65.287
	-6.175	61.454
	-6.175	-61.454
	-170.08	0.0
	-209.32	0.0
	-210.51	0.0
	-89841.	0.0
	-90155.	0.0
	-91513.	0.0

To assess the robustness of the feedback control law, the feedback gain matrix calculated at the design flight condition with [Q] equal to the zero matrix was applied at off-design flight conditions assuming full state feedback. Contained in Table 3 are the closed loop eigenvalues at representative off-design velocities. As shown by this table and in Figure 3, this gain matrix does perform adequately up to the design velocity of 187 fps and slightly beyond, but narrowly maintains stability at 156 fps. Above approximately 194 fps, the torsion mode becomes unstable but does recross the imaginary axis into the stable regime at higher velocities. This is not an unusual result as the control law is optimized for one flight condition and does not guarantee acceptable performance at any other flight condition. The fact that the closed loop system is stable up to the design velocity is an excellent result, but to achieve robustness throughout the complete flight regime this control would not be acceptable.

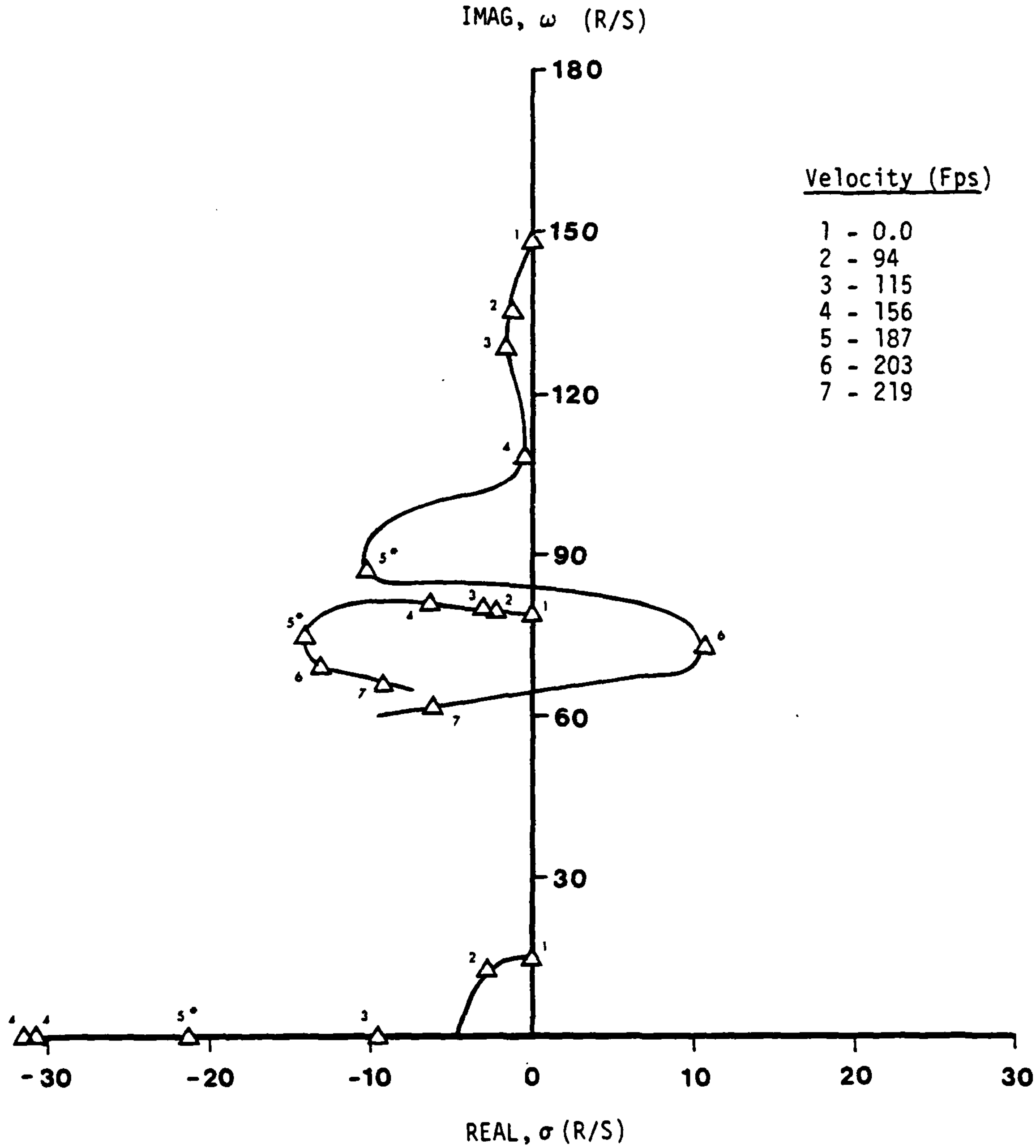


Figure 3. Closed Loop Velocity Root Locus Assuming Full State Feedback Gains

An attempt was then made to vary [Q] and the individual state weightings inside [Q] to obtain a feedback gain matrix that exhibits satisfactory performance for velocities greater than 187 fps. Since it is the torsion mode that becomes unstable at higher velocities, the state associated with that mode was weighted more heavily inside the weighting matrix [Q]. This has the effect of directing more of the available control authority to controlling this mode of instability. After several iterations, a weighting of 100 on the torsion mode and 0.1 on the other modes produced more desirable results as shown in Figure 4. This feedback control law exhibited satisfactory performance for all of the velocities investigated.

It was not the intent of this thesis to assess whether this closed loop performance is acceptable when considering the actual dynamic response of the aircraft when disturbed from a steady state condition. It should be understood that this study involved standard eigenvalue analysis techniques, and the criterion to judge acceptable performance was only that the eigenvalues have negative real parts, i.e. the system, when excited from a steady state condition, would have a transient response which decays to zero over time.

As discussed earlier, the full state vector is not available for measurement and therefore an observer (state estimator) must be designed to provide an estimate of the state vector to be used in the feedback control law. The observer gains were determined using the diagonalized state coefficient matrix and the transpose of the output coefficient matrix which were input into the modified OPTCON program, and the weighting matrices varied to obtain the desired observer poles. Once again,

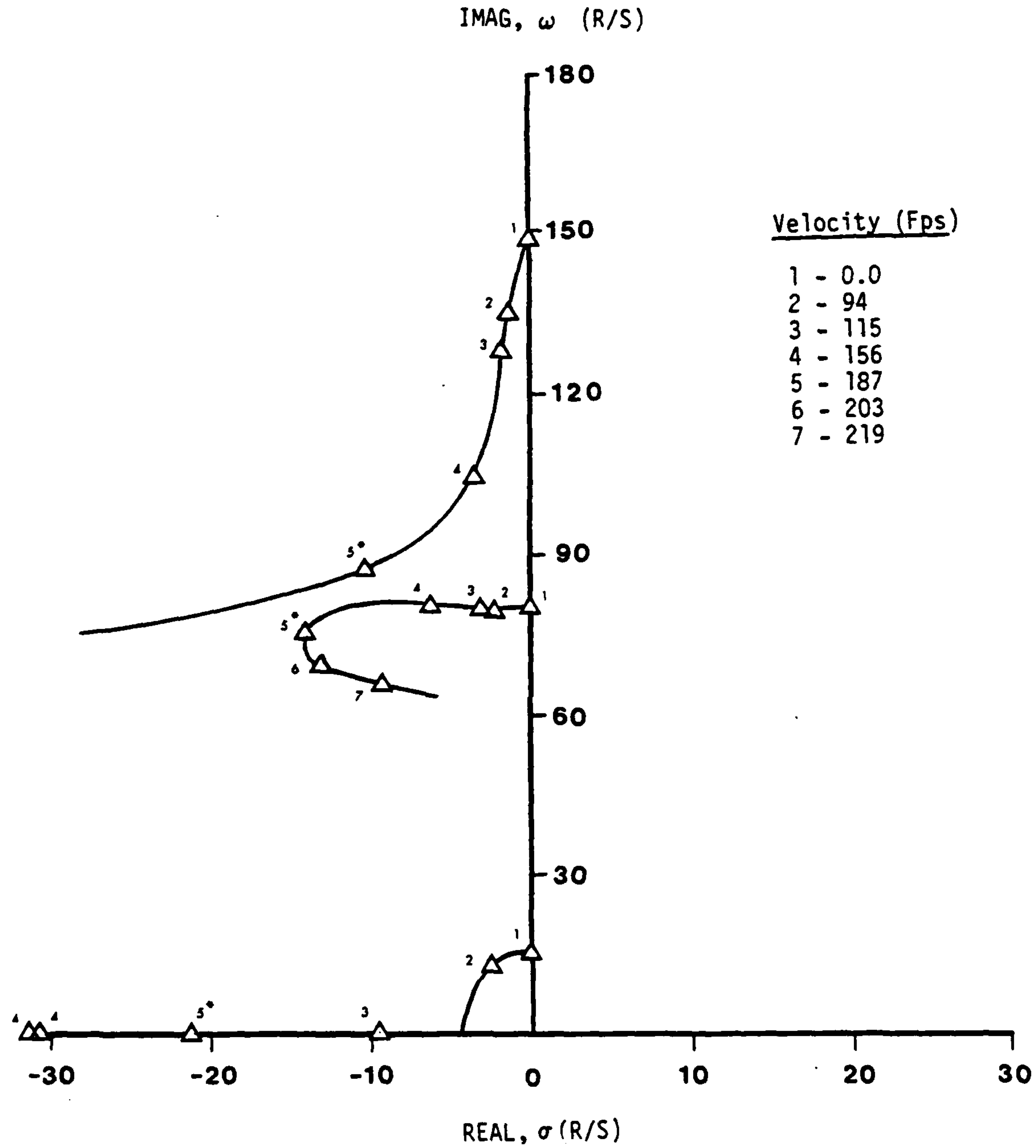


Figure 4. Closed Loop Velocity Root Locus With Improved Weightings.

the control weighting matrix $[R]_{OB}$ was kept equal to the identity matrix and only $[Q]_{OB}$ was varied. After several iterations, the largest weighting matrix that could be utilized was $[Q]_{OB} = [I]$ due to numerical difficulties inside the OPTCON program. It was decided that although the observer poles were not as deep in the left half plane as was desired, the eigenvalues obtained were adequate for the current analysis and for demonstration of optimal control theory methodology.

The observer poles obtained for this case at the design velocity are presented in Table 4 and can be seen to be adequately stable, but not

Table 4

Observer Poles for Design Velocity
$[R]_{OB} = [Q]_{OB} = [I]$

No	Real	Imaginary
1	-25.257	0.0
2	-62.662	0.0
3	-14.943	75.729
4	-14.943	-75.729
5	-10.943	87.413
6	-10.943	-87.413
7	-158.52	0.0
8	-188.38	0.0
9	-3846.3	0.0
10	-76714.	0.0
11	-76969.	0.0
12	-78118.	0.0

as rapidly convergent as was desired. As with the closed loop eigenvalues, these observer poles would vary as velocity changes, and improved weighting matrices might be required. This was not investigated in this study due to the numerical problems encountered in the observer design. In the

procedure to design an observer to be actually implemented, it might be warranted to schedule the observer gain matrix as a function of dynamic pressure so as to assure fast convergence of the reconstruction error to zero throughout the flight envelope.

One other area of interest was directed at determining the effect of neglecting the three high frequency aerodynamic states in the control law synthesis. This results in a reduced order state model (9th instead of 12th) which simplifies the numerical calculations and improves the mathematical inaccuracies. When the state model was formed, the feedback gain matrix calculated and the closed loop eigenvalues determined, it was discovered that neglecting these poles had no noticable effect on the closed loop performance (see Table 5). This is as would be expected

Table 5

Reduced Order Model, Closed Loop Eigenvalues
At Design Velocity

No	Real	Imaginary
1	-21.190	0.0
2	-57.342	0.0
3	-14.034	74.807
4	-14.034	-74.807
5	-10.393	87.113
6	-10.393	-87.113
7	-154.03	0.0
8	-178.77	0.0
9	-179.75	0.0

due to the large magnitude of the neglected poles.

This is also a very interesting result as it illustrates that so-

phisticated, three dimensional unsteady aerodynamic modeling techniques can be utilized to form the aeroelastic equations of motion, but subsequently the equations associated with the high frequency aerodynamic lag states can be neglected. This has the effect of reducing the order of the state model formed and thereby reducing the complexity of the numerical operations performed, and eliminates possible numerical difficulties inside the Ricatti equation solver subroutine.

IV. Conclusions and Recommendations

It has been demonstrated herein that optimal control theory techniques can be applied adequately for the prevention of aeroelastic instabilities on a forward swept wing aircraft. The feedback control law synthesized at the chosen design point performed adequately throughout the flight regime. It was also illustrated that an observer could be properly designed to reconstruct the original state vector from available sensor measurements. The feedback control law was also formed from a reduced order state model which neglected the equations associated with the high frequency aerodynamic lag states. The control law synthesized from this state model performed similarly for all velocities investigated, and the resulting closed loop eigenvalues were not altered by any noticeable amount.

The results of this study are just a foundation for an area of control system design that warrants further investigation. It is planned in in the near future to expand this study to include similar analyses for the forward swept wing model free in pitch and subsequently for the model free in pitch and plunge. For the model free in pitch, coupling between the wing first bending mode and the rigid body pitch mode occurs at a velocity lower than the static divergence speed, and therefore becomes the instability of interest. To complete the analysis of the forward swept wing model, the case for the model free in pitch and plunge will be investigated so as to actively control the more realistic representation of an actual aircraft employing a forward swept wing design.

There are several additional concepts that could be applied to this study to further enhance the validity of the results. Although it is realized that control surface displacements and rates are important in the design and effectiveness of a control law, the calculation of these quantities was not considered in this study. Also, sensor and actuator dynamics were neglected in this analysis. The control of divergence for the cantilever wing and the body freedom flutter instability for the model free in pitch would be expected to be insensitive to actuator and sensor dynamics because of the low frequencies involved. However, it is anticipated that the control of the higher frequency wing bending/torsion flutter mode would be affected by the addition of these components.

Another possible area of investigation might be to perform a gust analysis on the closed loop control system to obtain a more robust control law which is adequate throughout the flight envelope. Several techniques have been investigated such as injecting white noise into the controller system model at the point of entry of the control inputs, during the process of tuning the filter (observer). This would allow the controller to exhibit more robustness at the design points, which may reduce the number of required design conditions necessary to have adequate performance throughout the flight envelope. It may even be desirable to design a fixed controller which is sufficiently robust to produce acceptable performance for all conditions in the aircraft's flight envelope. A natural extension to this is to consider injecting time-correlated noise. This allows the robustification technique to be applied only over a desired frequency range rather than over all frequencies as in the previous method.

Bibliography

1. Wykes, John H. and R.J. Knight. "Progress Report On A Gust Alleviation And Structural Dynamic Stability Augmentation System (GASDSAS) Design Study," AIAA Paper No. 66-999 (December 1966).

2. Burris, P.M., J.B. Dempster and R.P. Johannes. "Flight Testing Structural Performance of the LAMS Flight Control System," AIAA Paper No. 68-244 (March 1968).

3. Wykes, John H. "Structural Dynamic Stability Augmentation and Gust Alleviation of Flexible Aircraft," AIAA Paper No. 68-1067 (October 1968).

4. Wykes, John H. and A.S. Mori. "An Analysis of Flexible Aircraft Structural Mode Control," AFFDL-TR-65-190 (June 1966).

5. Griffen, Kenneth E. and Franklin E. Eastep. "Active Control of Forward Swept Wings with Divergence and Flutter Aeroelastic Instabilities," AIAA Paper No. 81-0637.

6. Chipman, Robert R., Alex M. Zislin and Catherine Waters. "Active Control of Aeroelastic Divergence," AIAA Paper No. 82-0684 (May 1982).

7. Miller, Gerald D., John H. Wykes and Michael J. Brosnan. "Rigid Body-Structural Mode Coupling on a Forward Swept Wing Aircraft," AIAA Paper No. 82-0683 (May 1982).

8. Weisshaar, T.A., T.A. Zeiler, T.J. Hertz and M.H. Shirk. "Flutter of Forward Swept Wings, Analysis and Test," Paper presented at the 23rd AIAA SDM Conference, New Orleans, LA (April 1982).

9. Noll, Thomas E., Franklin E. Eastep and Robert A. Calico. "Active Suppression of Aeroelastic Instabilities on a Forward Swept Wing," Journal of Aircraft, 3: 202-208 (March 1984).

10. Geising, J.P., T.P. Kalman and W.P. Rodden. "Subsonic Unsteady Aerodynamics for General Configurations," AFFDL-TR-75-5 Part I, Vols. I and II (November 1971).

11. Kwakernaak, Huibert and Raphael Sivan. Linear Optimal Control Systems. New York: John Wiley and Sons, Inc., 1972.

12. Bisplinghoff, R. and Holt Ashley. Principles of Aeroelasticity. New York: John Wiley and Sons, Inc., 1975.

13. Floyd, Robert M. Doctoral Student (personal interviews). Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, September 1983 - June 1984.

14. Fortmann, Thomas E. and Konrad L. Hitz. An Introduction to Linear Control Systems. New York: Marcel Dekker, Inc., 1977.

15. Ogata, Katsuhiko. State Space Analysis of Control Systems. Englewood Cliffs: Prentice-Hall Inc., 1967.

16. D'Azzo, John J. and Constantine H. Houpis. Linear Control System Analysis and Design (second edition). New York: McGraw-Hill Book Company, 1975.

17. Newsom, Jerry R. "Control Law Synthesis for Active Flutter Suppression Using Optimal Control Theory." Journal of Guidance and Control, 2: 388-394 (May 1979).

18. ——. "A Method for Obtaining Practical Flutter Suppression Control Laws Using Results of Optimal Control Theory," NASA-TP-1471 (August 1979).

19. Mordechay, Karpel. "Design for Active and Passive Flutter Suppression and Gust Alleviation," NASA-CR-3482 (November 1981).

20. Wilt, Mirmak, and Ray. "OPTCON - Solution of Linear Quadratic Optimal Control Problems," Master Thesis, Air Force Institute of Technology, 1972.

21. Strang, Gilbert. Linear Algebra and its Applications. New York : Academic Press Inc., 1976.

22. Vepa, R. "On the use of Pade Approximants to Represent Unsteady Aerodynamic Loads for Arbitrary Small Motions of Wings," AIAA Paper No. 76-17 (January 1976).

Appendix A: Aeroelastic Equations of Motion

The aeroelastic equations of motion of a flexible aircraft in an airstream can be represented as (Ref 12):

$$[M]\ddot{\underline{q}}(t) + [C]\dot{\underline{q}}(t) + [K]\underline{q}(t) = \underline{F}(t) \qquad \text{(A-1)}$$

where [M], [C] and [K] are the generalized mass, damping and stiffness matrices obtained using a set of generalized coordinates $\underline{q}(t)$ and several natural vibration mode shapes, and $\underline{F}(t)$ represents the unsteady aerodynamic forces. In Noll's study (Ref 9), the forces were obtained from subsonic doublet lattice unsteady aerodynamic theory (Ref 10) and are defined to be

$$\underline{F}(t) = -1/2\rho V^2 \tilde{s}[Q(k)]\underline{q}(t) \qquad \text{(A-2)}$$

where $\tilde{s}$ represents a reference area.

The generalized aerodynamic force coefficient matrix [Q(k)] was computed from

$$Q_{ij} = \iint \frac{\Delta P_j}{1/2\rho V^2} \; \frac{h_i}{\tilde{s}} \, dxdy \qquad \text{(A-3)}$$

where h_i is the normalized vertical displacement in the ith mode. The

coefficient Q_{ij} represents the non-dimensional modal force in the ith mode due to pressure from the jth mode and is dependent on the reduced frequency k (where $k = b\omega/V$). The expression for $\underline{F}(t)$ can now be substituted into Eq A-1 and the Laplace Transform taken to yield (for zero initial conditions):

$$([M]s^2 + [C]s + [K] + 1/2\rho V^2 \tilde{s}[Q(s)])\underline{q}(s) = 0 \qquad \text{(A-4)}$$

Before the equations of motion can be transformed into the time domain, it is necessary to express the elements of the [Q(s)] matrix in a convenient form. For the current analysis, it was assumed that the elements of [Q(s)] are given by

$$Q_{ij} = \frac{N_0 + N_1\bar{s} + N_2\bar{s}^2 + N_3\bar{s}^3}{1 + D_1\bar{s} + D_2\bar{s}^2} \qquad \text{(A-5)}$$

where

$$\bar{s} = bs/V$$

which is a Pade' approximant representation of the aerodynamic forces (Ref 22). The coefficients N_i and D_i were obtained using a least squares fitting process over a reduced frequency range of interest. The generalized force matrix now becomes (in terms of the Laplace variable s):

$$[Q(s)] = \frac{[C_0]V^3 + [C_1]V^2bs + [C_2]Vb^2s^2 + [C_3]b^3s^3}{V^3 + D_1bV^2s + D_2Vb^2s} \qquad \text{(A-6)}$$

Substituting this expression into Eq A-4 and multiplying through by the denominator polynomial, the transformed equations of motion take the form:

$$([A_4]s^4 + [A_3]s^3 + [A_2]s^2 + [A_1]s + [A_0])\underline{q}(s) = 0 \qquad \text{(A-7)}$$

The elements of the $[A_i]$ matrices are functions of velocity and therefore the roots of the characteristic equation can now be determined as velocity is varied.

When control surfaces are added to the flexible wing, their effect must be taken into consideration when forming the equations of motion. We can begin the derivation by incorporating the effects of control surfaces into Eq A-1 :

$$[M]\ddot{\underline{q}}(t) + [C]\dot{\underline{q}}(t) + [K]\underline{q}(t) + 1/2\rho V^2\tilde{s}[Q(k)]\underline{q}(t) + [M_c]\ddot{\underline{q}}_c(t) + 1/2\rho V^2\tilde{s}[Q_c(k)]\underline{q}_c(t) = 0 \ , \qquad \text{(A-8)}$$

where $[M_c]$ and $[Q_c]$ are matrices of order NM x NC (NM equals the number of elastic modes and NC is the number of control surfaces). For this study, two active control surfaces were employed (leading and trailing edge) resulting in

$$\underline{q}_c(t) = [\,\delta_{TE}(t) \quad \delta_{LE}(t)]^T \qquad \text{(A-9)}$$

The $[Q_c(s)]$ matrix to be used in the Laplace Transform of Eq A-8 is also obtained using a Pade' approximant for the control surface aerodynamics

and takes on the same form as Eq A-6 :

$$[Q_c(s)] = \frac{[C_{0c}]V^3 + [C_{1c}]V^2bs + [C_{2c}]Vb^2s^2 + [C_{3c}]b^3s^3}{V^3 + D_1bV^2s + D_2Vb^2s^2} \qquad \text{(A-10)}$$

The denominator of the $[Q_c(s)]$ matrix is also constrained to be the same as the denominator of the $[Q(s)]$ matrix.

Now, when the aerodynamic modeling is substituted into the Transform of Eq A-8 and the resulting equation is multiplied through by the denominator of Eqs A-6 and A-10, the aeroelastic equations of motion take on the form:

$$\begin{aligned}&([A_4]s^4 + [A_3]s^3 + [A_2]s^2 + [A_1]s + [A_0])\underline{q}(s)\\ &\quad + ([B_4]s^4 + [B_3]s^3 + [B_2]s^2 + [B_1]s + [B_0])\underline{q}_c(s) = 0 \qquad \text{(A-11)}\end{aligned}$$

where

$$\begin{aligned}
[A_4] &= [M]D_2b^2\\
[A_3] &= [M]D_1bV + [C]D_2b^2 + 1/2\rho\tilde{s}[C_3]Vb^3\\
[A_2] &= [M]V^2 + [C]D_1bV + [K]D_2b^2 + 1/2\rho\tilde{s}[C_2]V^2b^2\\
[A_1] &= [C]V^2 + 1/2\rho\tilde{s}[C_1]V^3b + [K]D_1bV\\
[A_0] &= [K]V^2 + 1/2\rho\tilde{s}[C_0]V^4
\end{aligned}$$

and (A-12)

$$\begin{aligned}
[B_4] &= [M_c]D_2b^2\\
[B_3] &= [M_c]D_1bV + 1/2\rho\tilde{s}[C_{3c}]Vb^3\\
[B_2] &= [M_c]V^2 + 1/2\rho\tilde{s}[C_{2c}]V^2b^2\\
[B_1] &= 1/2\rho\tilde{s}[C_{1c}]V^3b\\
[B_0] &= 1/2\rho\tilde{s}[C_{0c}]V^4
\end{aligned}$$

For this study, three elastic modes were utilized (first wing bending, second wing bending and first wing torsion) resulting in three generalized coordinates comprising the $\underline{q}(t)$ vector. The generalized coordinates correspond to displacements of the wing surface due to each of the elastic modes. $q_1(t)$ and $q_2(t)$ are displacements measured perpendicular to the elastic axis in ft, and $q_3(t)$ is the rotation about the elastic axis in radians. With $\underline{q}(t)$ being 3x1 and $\underline{q}_c(t)$ being 2x1, the resulting $[A_i]$ and $[B_i]$ matrices defined in Eq A-12 become 3x3 and 3x2 respectively. In addition, the aerodynamic control surfaces were assumed to be rigid, massless plates resulting in $[M_c]$ being a zero matrix. Eq A-11 can now be manipulated so that a state model can easily be obtained. By equating derivatives with powers of s (to revert back to time domain), multiplying through by $[A_4]^{-1}$ and shifting the control terms to the right hand side of the equation, the equations of motion become:

$$\ddddot{\underline{q}}(t) + [a_3]\dddot{\underline{q}}(t) + [a_2]\ddot{\underline{q}}(t) + [a_1]\dot{\underline{q}}(t) + [a_0]\underline{q}(t)$$
$$= [b_3]\dddot{\underline{q}}_c(t) + [b_2]\ddot{\underline{q}}_c(t) + [b_1]\dot{\underline{q}}_c(t) + [b_0]\underline{q}_c(t) \qquad \text{(A-13)}$$

where

$$\begin{aligned} [a_3] &= [A_4]^{-1}[A_3] & [b_3] &= -[A_4]^{-1}[B_3] \\ [a_2] &= [A_4]^{-1}[A_2] & [b_2] &= -[A_4]^{-1}[B_2] \\ [a_1] &= [A_4]^{-1}[A_1] & [b_1] &= -[A_4]^{-1}[B_1] \\ [a_0] &= [A_4]^{-1}[A_0] & [b_0] &= -[A_4]^{-1}[B_0] \end{aligned} \qquad \text{(A-14)}$$

Eq A-13 can now be cast into state space form by making the following

substitutions (Ref 15):

$$
\begin{aligned}
\underline{X}_1(t) &= \underline{q}(t) \\
\underline{X}_2(t) &= \dot{\underline{X}}_1(t) - [c_1]\underline{u}(t) \\
\underline{X}_3(t) &= \dot{\underline{X}}_2(t) - [c_2]\underline{u}(t) \\
\underline{X}_4(t) &= \dot{\underline{X}}_3(t) - [c_3]\underline{u}(t) \\
\dot{\underline{X}}_4(t) &= -[a_0]\underline{X}_1(t) - [a_1]\underline{X}_2(t) - [a_2]\underline{X}_3(t) \\
&\qquad - [a_3]\underline{X}_4(t) + [c_4]\underline{u}\ (t)
\end{aligned}
\tag{A-15}
$$

where

$$
\begin{aligned}
[c_1] &= [b_3] \\
[c_2] &= [b_2] - [a_3][c_1] \\
[c_3] &= [b_1] - [a_2][b_3] - [a_3][c_2] \\
[c_4] &= [b_0] - [a_1][b_3] - [a_2][c_2] - [a_3][c_3]
\end{aligned}
$$

Employing these substitutions, Eq A-13 can be cast into the following state space representation :

$$
\begin{aligned}
\dot{\underline{X}}(t) &= [A]\underline{X}(t) + [B]\underline{u}(t) \\
\underline{Y}(t) &= [C]\underline{X}(t)
\end{aligned}
\tag{A-16}
$$

where the state coefficient matrix [A] is in companion form and the control coefficient matrix [B] is a full matrix, that is:

$$[A] = \begin{bmatrix} 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ -a_0 & -a_1 & -a_2 & -a_3 \end{bmatrix} \quad \text{and} \quad [B] = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

In this study, two sensors were utilized resulting in the measurable outputs being the wing vertical displacement h and angle of twist α. The resulting output vector $\underline{Y}(t)$ is 2x1 and is expressed as a combination of the generalized coordinates, that is :

$$y_j(t) = \sum_{i=1}^{NM} \phi_{ij} \, \underline{q}_i(t) \qquad \text{(A-17)}$$

where NM is the number of elastic modes. The output coefficient matrix [C] then takes on the form,

$$[C] = [\ C' \ : \ 0\ ] \qquad \text{(A-18)}$$

where

$$C' = \begin{bmatrix} \text{---} \ \phi_1 \ \text{---} \\ \text{---} \ \phi_2 \ \text{---} \end{bmatrix}$$

$$= \begin{bmatrix} -.00189 & .03043 & .60840 \\ .70683 & .08840 & .68922 \end{bmatrix}$$

and

0 = a 2x9 sub-matrix of zeroes

which results in

$$\underline{Y}(t) = [C]\underline{X}(t) = \begin{Bmatrix} h(t) \\ \alpha(t) \end{Bmatrix} . \qquad \text{(A-20)}$$

Appendix B: State Model Formulation/Root Locus Program

Contained on the following pages is a listing of the program RLOCUS utilized to form the state coefficient matrices as a function of free-stream velocity and density. A sample input file is also given at the end of the Appendix. The Pade' polynomials used as partial input to this program were formed using the program PADE written by Thomas Noll. To run this program on the CYBER 750/74, the following commands should be executed:

```
/ATTACH,LINPACK/UN=APPLIB.
/ATTACH,EISPACK/UN=APPLIB.
/LIBRARY,LINPACK,EISPACK.
/GET,(INPUT FILE).
/GET,LGO=RLOCUS/UN=E710067.
/LGO,(INPUT FILE),DATA.
```

The output is then contained in local file DATA which may be cataloged by the user for further use.

```
      PROGRAM RLOCUS(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
C
C     PROGRAM USED FOR CANTILEVER OR PITCH ANALYSES
C        ALL DIMENSIONS ARE IN SLUGS AND FT
C        NMODE      NUMBER OF GENERALIZED COORDINATES
C        NSURF      NUMBER OF ACTIVE CONTROL SURFACES (LE 2)
C
      DIMENSION C(4,4),B0(4,4),B1(4,4),B2(4,4),B3(4,4),AI(4,4)
      DIMENSION G(161),W(161),VEL1(161),VGA(960),OMEGA(48)
      DIMENSION AMAT(48,48),BMAT(48,48),BM0(4,4),BM1(4,4),BM2(4,4)
      DIMENSION BM3(4,4),BM4(4,4),Z4(4,4),Z3(4,4),Z2(4,4),Z1(4,4)
      DIMENSION Z0(4,4),Z3A(4,4),Z2A(4,4),Z1A(4,4),Z0A(4,4)
      DIMENSION Z3B(4,4),Z2B(4,4),Z1B(4,4),Z0B(4,4),Z1C(4,4)
      DIMENSION Z0C(4,4),Z0D(4,4)
      DIMENSION SR2(48),SI2(48),IV2(48),FV2(48),OMEG(48)
      REAL M,K(4,4),MC(4,4)
      COMMON /PL/ JV,IV,XA(960),YA(960)
      COMMON /VEL/ CM0(6,6),CM1(6,6),CM2(6,6),CM3(6,6),DM0,DM1,
     1CM2,V
      COMMON /ACT/ A0(4,4),A1(4,4),A2(4,4),A3(4,4),A4(4,4),M(4,4),
     1C0(6,6),C1(6,6),C2(6,6),C3(6,6),IS,IPR1,IPR2,XSCALE
      COMMON /EIGEN/ A(48,48),SR(48),SI(48),SR1(48),SI1(48),
     1FV1(48),IV1(48),SIHZ(48)
      JV = 720
      IV = 0
      IS = 12
      ISMAX = 48
      IVMAX = 15
      ITOTP = 16*IVMAX
      ITOTA = ISMAX*IVMAX+ITOTP
      DO 33 I=1,ITOTA
   33 XA(I) = YA(I) = VGA(I) = 0.0
      DO 41 I=1,ISMAX
   41 SR(I) = SI(I) = 0.0
      DO 45 I=1,48
      DO 45 J=1,48
      A(I,J)=0.0
      AMAT(I,J)=0.0
      BMAT(I,J)=0.0
   45 CONTINUE
C
C
      DO 46 I=1,4
      DO 46 J=1,4
      BM0(I,J)=0.0
      BM1(I,J)=0.0
      BM2(I,J)=0.0
      BM3(I,J)=0.0
      BM4(I,J)=0.0
      Z0(I,J)=0.0
      Z1(I,J)=0.0
```

```
      Z2(I,J)=0.0
      Z3(I,J)=0.0
      Z4(I,J)=0.0
      Z0A(I,J)=0.0
      Z1A(I,J)=0.0
      Z2A(I,J)=0.0
      Z3A(I,J)=0.0
 46   CONTINUE
C
C     TIME SCALING FACTOR
C
C********************
      BETA=0.001
C********************
C
      BET1=BETA
      BET2=BET1*BETA
      BET3=BET2*BETA
      BET4=BET3*BETA
C
      NPASS = 1
      WRITE(6,3)
C READ IN DATA FOR PASSIVE FLUTTER ANALYSIS
C NUMBER OF MODES, SOLUTION TYPE AND NUMBER OF ACTIVE SURFACES
C PRINT COMMANDS  IPR1 = 0 WRITES ALL INPUT DATA
C                 IPR2 = 0 WRITES ALL EIGENVALUES
C                 IPR1 = 1 NO WRITE
C                 IPR2 = 1 WRITES ONLY SIGNIFICANT EIGENVALUES
C
C     NSOLTYP READ IN AS ZERO SINCE ACTIVE CONTROL
C     ANALYSIS IS DONE BY SEPARATE ANALYSIS
C
      READ(5,1) NMODE,NSOLTYP,NSURF,IPR1,IPR2
C REFERENCE CHORD, AIR DENSITY, REFERENCE SEMICHORD AND PLOT
C SCALING PARAMETERS IN X-DIRECTION AND Y-DIRECTION
      READ(5,5) BREF,RHO,SREF,XSCALE,YSCALE
      WRITE(6,14)
      IF(NSOLTYP.EQ.0) WRITE(6,15)
      IF(NSOLTYP.EQ.1) WRITE(6,16)
      IF(NSOLTYP.EQ.2) WRITE(6,17)
      WRITE(6,18) NMODE,NSURF
      WRITE(6,19) BREF,RHO,SREF
      DO 20 I=1,NMODE
      DO 21 J=1,NMODE
      MC(I,J)=0.0
 21   M(I,J) = C(I,J) = K(I,J) = 0.0
 20   CONTINUE
      CONST = .5*RHO*SREF
C GENERALIZED MASS
      READ(5,5) (M(I,I),I=1,NMODE)
C GENERALIZED DAMPING
      READ(5,5) (C(I,I),I=1,NMODE)
```

```
C GENERALIZED STIFFNESS
      READ(5,5) (K(I,I),I=1,NMODE)
      IF(IPR1.EQ.1) GO TO 39
      WRITE(6,3)
      WRITE(6,4)
      DO 24 I=1,NMODE
 24   WRITE(6,2) (M(I,J),J=1,NMODE)
      WRITE(6,6)
      DO 22 I=1,NMODE
 22   WRITE(6,2) (C(I,J),J=1,NMODE)
      WRITE(6,7)
      DO 23 I=1,NMODE
 23   WRITE(6,2) (K(I,J),J=1,NMODE)
 39   CONTINUE
      NM = NMODE+NSURF
C PADE POLYNOMIALS OF AERODYNAMIC FORCES FOR THE ELASTIC
C MODES AND ALL CONTROL SURFACES
      READ(5,5) ((C0(I,J),J=1,NM),I=1,NM)
      READ(5,5) ((C1(I,J),J=1,NM),I=1,NM)
      READ(5,5) ((C2(I,J),J=1,NM),I=1,NM)
      READ(5,5) ((C3(I,J),J=1,NM),I=1,NM)
      READ(5,5) D1,D2
      D0 = 1.0
      IF(IPR1.EQ.1) GO TO 81
      WRITE(6,3)
      WRITE(6,8)
      DO 28 I=1,NMODE
      DO 29 J=1,NMODE
      WRITE(6,9) I,J
 29   WRITE(6,10) C0(I,J),C1(I,J),C2(I,J),C3(I,J),D0,D1,D2
 28   CONTINUE
 81   CONTINUE
C VELOCITY VARIATION
C*******************************************************
 100  CALL VEL(C0,C1,C2,C3,D0,D1,D2,NM,BREF,CONST,NPASS)
C*******************************************************
      CALL WRITE6(C0,NM,NM)
      CALL WRITE6(C1,NM,NM)
      CALL WRITE6(C2,NM,NM)
      CALL WRITE6(C3,NM,NM)
      WRITE(6,87) D0,D1,D2
 87   FORMAT(//,1X,"D0,D1,D2 =",3E12.5,/)
      IF(V.LT.0.0) GO TO 25
      WRITE(6,89) DM0,DM1,DM2
 89   FORMAT(//,1X,"DM0,DM1,DM2 =",3E12.5,/)
      CALL WRITE6(CM0,NM,NM)
      CALL WRITE6(CM1,NM,NM)
      CALL WRITE6(CM2,NM,NM)
      CALL WRITE6(CM3,NM,NM)
      DO 30 I=1,NMODE
C FORM EIGENVALUE PROBLEM
      DO 31 J=1,NMODE
```

```
      A4(I,J) = DM2*M(I,J)
      AI(I,J) = A4(I,J)
      A3(I,J) = DM1*M(I,J)+DM2*C(I,J)+CM3(I,J)
      A2(I,J) = DM0*M(I,J)+DM1*C(I,J)+DM2*K(I,J)+CM2(I,J)
      A1(I,J) = DM0*C(I,J)+DM1*K(I,J)+CM1(I,J)
 31   A0(I,J) = DM0*K(I,J)+CM0(I,J)
 30   CONTINUE
      CALL WRITE4(A0,NMODE,NMODE)
      CALL WRITE4(A1,NMODE,NMODE)
      CALL WRITE4(A2,NMODE,NMODE)
      CALL WRITE4(A3,NMODE,NMODE)
      CALL WRITE4(A4,NMODE,NMODE)
      CALL WRITE4(AI,NMODE,NMODE)
      IF(NSOLTYP.EQ.2.AND.NPASS.EQ.2) GO TO 103
      CALL INVERT(AI,NMODE)
      CALL WRITE4(AI,NMODE,NMODE)
      CALL MULTMM(AI,A0,B0,NMODE,NMODE,NMODE)
      CALL MULTMM(AI,A1,B1,NMODE,NMODE,NMODE)
      CALL MULTMM(AI,A2,B2,NMODE,NMODE,NMODE)
      CALL MULTMM(AI,A3,B3,NMODE,NMODE,NMODE)
      CALL WRITE4(B0,NMODE,NMODE)
      CALL WRITE4(B1,NMODE,NMODE)
      CALL WRITE4(B2,NMODE,NMODE)
      CALL WRITE4(B3,NMODE,NMODE)
C
C
      DO 50 I=1,NMODE
      DO 51 J=1,NSURF
      JJ=NMODE+NSURF-J+1
      BM0(I,J)=CM0(I,JJ)
      BM1(I,J)=CM1(I,JJ)
      BM2(I,J)=MC(I,J)*DM0+CM2(I,JJ)
      BM3(I,J)=MC(I,J)*DM1+CM3(I,JJ)
      BM4(I,J)=MC(I,J)*DM2
 51   CONTINUE
 50   CONTINUE
C
      CALL WRITE4(BM0,NMODE,NSURF)
      CALL WRITE4(BM1,NMODE,NSURF)
      CALL WRITE4(BM2,NMODE,NSURF)
      CALL WRITE4(BM3,NMODE,NSURF)
      CALL WRITE4(BM4,NMODE,NSURF)
C
      CALL MULTMM(AI,BM4,Z4,NMODE,NMODE,NSURF)
      CALL MULTMM(AI,BM3,Z3,NMODE,NMODE,NSURF)
      CALL MULTMM(AI,BM2,Z2,NMODE,NMODE,NSURF)
      CALL MULTMM(AI,BM1,Z1,NMODE,NMODE,NSURF)
      CALL MULTMM(AI,BM0,Z0,NMODE,NMODE,NSURF)
C
      CALL MULTMM(B3,Z3,Z2A,NMODE,NMODE,NSURF)
      CALL MULTMM(B2,Z3,Z1A,NMODE,NMODE,NSURF)
      CALL MULTMM(B1,Z3,Z0A,NMODE,NMODE,NSURF)
```

```
C
      CALL WRITE4(Z4,NMODE,NSURF)
      CALL WRITE4(Z3,NMODE,NSURF)
      CALL WRITE4(Z2,NMODE,NSURF)
      CALL WRITE4(Z1,NMODE,NSURF)
      CALL WRITE4(Z0,NMODE,NSURF)
C
      CALL WRITE4(Z3A,NMODE,NSURF)
      CALL WRITE4(Z2A,NMODE,NSURF)
      CALL WRITE4(Z1A,NMODE,NSURF)
      CALL WRITE4(Z0A,NMODE,NSURF)
C
C     FORM BMAT FOR STATE MODEL
C
      I1=NMODE
      I2=2*NMODE
      I3=3*NMODE
C
      DO 42 I=1,NMODE
      DO 42 J=1,NSURF
      Z3B(I,J) = -Z3(I,J)
      Z2B(I,J) = -Z2(I,J) + Z2A(I,J)
 42   CONTINUE
      CALL MULTMM(B3,Z2B,Z1C,NMODE,NMODE,NSURF)
      CALL WRITE4(Z1C,NMODE,NSURF)
      DO 43 I=1,NMODE
      DO 43 J=1,NSURF
      Z1B(I,J) = -Z1(I,J) + Z1A(I,J) - Z1C(I,J)
 43   CONTINUE
      CALL MULTMM(B2,Z2B,Z0C,NMODE,NMODE,NSURF)
      CALL MULTMM(B3,Z1B,Z0D,NMODE,NMODE,NSURF)
      CALL WRITE4(Z0C,NMODE,NSURF)
      CALL WRITE4(Z0D,NMODE,NSURF)
      DO 44 I=1,NMODE
      DO 44 J=1,NSURF
      Z0B(I,J) = -Z0(I,J) + Z0A(I,J) - Z0C(I,J) - Z0D(I,J)
 44   CONTINUE
C
      DO 53 I=1,NMODE
      DO 54 J=1,NSURF
      BMAT(I,J)=BET1*Z3B(I,J)
      BMAT(I1+I,J)=BET2*Z2B(I,J)
      BMAT(I2+I,J)=BET3*Z1B(I,J)
      BMAT(I3+I,J)=BET4*Z0B(I,J)
 54   CONTINUE
 53   CONTINUE
C
C
C FORM UNAUGMENTED DYNAMIC MATRIX
      N1 = NMODE*4
      N2 = NMODE*3
C
```

```
      DO 82 I=1,N1
      DO 82 J=1,N1
 82   A(I,J)=0.0
C
      DO 36 I=1,N2
      J = NMODE+I
 36   A(I,J) = 1.0
      N3 = N2+1
      II = 0
      DO 37 I=N3,N1
      I1 = NMODE
      I2 = 2*NMODE
      I3 = 3*NMODE
      II = II+1
      JJ = 0
      DO 38 J=1,NMODE
      JJ = JJ+1
      A(I,J) = -B0(II,JJ)
      I1 = I1+1
      A(I,I1) = -B1(II,JJ)
      I2 = I2+1
      A(I,I2) = -B2(II,JJ)
      I3 = I3+1
 38   A(I,I3) = -B3(II,JJ)
 37   CONTINUE
C
C     FORM AMAT MATRIX FOR STATE MODEL
C
      DO 83 I=1,N1
      DO 83 J=1,N1
 83   AMAT(I,J)=0.0
C
      DO 70 I=1,N2
      J=NMODE+I
 70   AMAT(I,J) = 1.0
C
      N3=N2+1
      II=0
      DO 71 I=N3,N1
      I1=NMODE
      I2=2*NMODE
      I3=3*NMODE
      II=II+1
      JJ=0
      DO 72 J=1,NMODE
      JJ=JJ+1
      AMAT(I,J) = -B0(II,JJ)*BET4
      I1=I1+1
      AMAT(I,I1) = -B1(II,JJ)*BET3
      I2=I2+1
      AMAT(I,I2) = -B2(II,JJ)*BET2
      I3=I3+1
```

```
      AMAT(I,I3) = -B3(II,JJ)*BET1
   72 CONTINUE
   71 CONTINUE
      WRITE(6,95)
   95 FORMAT(//,1X,"AMAT FOR STATE MODEL :",/)
      DO 55 I=1,N1
   55 WRITE(6,98) (AMAT(I,J),J=1,10)
      WRITE(6,97)
      DO 56 I=1,N1
   56 WRITE(6,98) (AMAT(I,J),J=11,N1)
C
      WRITE(6,94)
   94 FORMAT(//,1X,"BMAT FOR STATE MODEL :",/)
      DO 57 I=1,N1
   57 WRITE(6,98) (BMAT(I,J),J=1,NSURF)
      WRITE(6,97)
C
      CALL RG(48,N1,AMAT,SR2,SI2,0,DUMM,IV2,FV2,IERR)
      WRITE(6,200)
  200 FORMAT(//,1X,"EIGENVALUES OF AMAT MATRIX :",//,6X,
     &"REAL",8X,"IMAG",8X,"OMEGA",/)
      DO 80 I=1,N1
      OMEG(I)=SQRT(SR2(I)**2+SI2(I)**2)
      WRITE(6,210) SR2(I),SI2(I),OMEG(I)
  210 FORMAT(1X,3E12.5)
   80 CONTINUE
C
C PERFORM EIGENVALUE SOLUTION (REQUIRES EISPACK ROUTINES)
      WRITE(6,99)
   99 FORMAT(//,1X," A MATRIX (IN MAIN) :",/)
      DO 90 I=1,N1
   90 WRITE(6,98) (A(I,J),J=1,10)
      WRITE(6,97)
   97 FORMAT(//)
   98 FORMAT(1X,10E12.5)
      DO 91 I=1,N1
   91 WRITE(6,98) (A(I,J),J=11,N1)
      CALL RG(48,N1,A,SR,SI,0,DUMMY,IV1,FV1,IERR)
      WRITE(6,11)
      PHI2 = 6.28318
      DO 40 I=1,N1
C CALCULATE ESTIMATE OF STRUCTURAL DAMPING
      GO = 0.0
      OMEGA(I)=SQRT(SR(I)**2+SI(I)**2)
      IF(SI(I).GT.1.E-2) GO = 2.*SR(I)/OMEGA(I)
      IF(ABS(SI(I)).LT.1.E-2) SI(I) = 0.0
      SIHZ(I)=SI(I)/PHI2
      JV = JV+1
      XA(JV) = SR(I)
      YA(JV) = SI(I)
      VGA(JV) = V
      IF(SI(I).EQ.0.0.AND.SR(I).EQ.0.0) GO TO 40
```

```
      IF(IPR2.EQ.1.AND.SI(I).EQ.0.0.AND.ABS(SR(I)).GT.XSCALE)
     1 GO TO 40
      WRITE(6,2) SR(I),SI(I),SIHZ(I),GO,OMEGA(I)
 40   CONTINUE
      WRITE(6,13)
      NPASS = 2
      GO TO 100
 25   CONTINUE
      DO 26 J=1,ITOTA
      IF(YA(J).LT.0.0) XA(J) = YA(J) = VGA(J) = 0.0
      IF(ABS(YA(J)).GT.YSCALE) XA(J) = YA(J) = VGA(J) = 0.0
 26   IF(ABS(XA(J)).GT.XSCALE) XA(J) = YA(J) = VGA(J) = 0.0
      IF(NSOLTYP.LE.1) CALL PLOT1
      IF(NSOLTYP.GE.1) CALL PLOT2
      IF(NSOLTYP.EQ.1) CALL PLOT3
      IG = 0
C CALCULATE STRUCTURAL DAMPING FOR PLOTTING
      DO 12 I=1,ITOTA
      YA(I) = YA(I)*PHI2
      IF(YA(I).EQ.0.0) GO TO 12
      IG = IG+1
      VEL1(IG) = VGA(I)
      W(IG) = SQRT(XA(I)**2+YA(I)**2)
      PS = XA(I)/W(IG)
      GO = 2.*PS
      DL = PHI2*PS/SQRT(1.-PS**2)
      PHI2S = PHI2**2
      G(IG) = 2.*PHI2*DL/(PHI2S-DL**2)
      IF(ABS(GO).GT.0.5) G(IG) = 0.0
      W(IG) = W(IG)/PHI2
 12   CONTINUE
C ESTABLISH ZERO FOR FREQUENCY VERSUS VELOCITY PLOT
      IG = IG+1
      W(IG) = G(IG) = VEL1(IG) = 0.0
      CALL PLOT4(G,W,VEL1,IG)
 1    FORMAT(5I5)
 2    FORMAT(10X,6E12.5)
 3    FORMAT(1H1)
 4    FORMAT(//,10X,*GENERALIZED MASS MATRIX*)
 5    FORMAT(6E12.5)
 6    FORMAT(//,10X,*GENERALIZED DAMPING MATRIX*)
 7    FORMAT(//,10X,*GENERALIZED STIFFNESS MATRIX*)
 8    FORMAT(//,10X,"HPADE POLYNOMIALS FOR AERODYNAMIC FORCES "
     1"(C0+C1*S+C2*S**2+C3*S**3/D0+D1*S+D2*S**2)")
 9    FORMAT(//,10X,*POLYNOMIAL FOR COEFFICIENT (*,I1,*,*,I1,*)*)
 10   FORMAT(10X,4E12.5)
 11   FORMAT(//,10X,"EIGENVALUES FOR PASSIVE SYSTEM :",//,12X,
     &"REAL(R/S)",3X,"IMAG(R/S)",4X,"IMAG(HZ)",4X,"DAMPING",
     &4X,"OMEGA(R/S)",/)
 13   FORMAT(10X,30H******************************)
 14   FORMAT(//,10X,*ROOT LOCUS FLUTTER ANALYSIS*)
 15   FORMAT(10X,*PASSIVE ANALYSIS ONLY*)
```

```
 16    FORMAT(10X,*PASSIVE AND ACTIVE CONTROL ANALYSES*)
 17    FORMAT(10X,*ACTIVE CONTROL ANALYSIS ONLY*)
 18    FORMAT(//,10X,*NUMBER OF MODES = *,I2,/10X,*NUMBER OF*
      1* CONTROL SURFACES = *,I2)
 19    FORMAT(//,10X,*REFERENCE SEMICHORD = *,E12.5* FT*/10X,
      1*AIR DENSITY = *,E12.5,12H SLUGS/FT**3/10X*REFERENCE *
      2*SEMISPAN*,E12.5* FT*= *)
       STOP
       END
C*********************************************************
       SUBROUTINE VEL(C0,C1,C2,C3,D0,D1,D2,N,B,CONST,NPASS)
C*********************************************************
C THIS ROUTINE ALLOWS FOR VELOCITY VARIATIONS FOR THE RLOCUS.
C NEGATIVE V STOPS THE SOLUTION PROCESS
       DIMENSION C0(6,6),C1(6,6),C2(6,6),C3(6,6)
       COMMON /VEL/ CM0(6,6),CM1(6,6),CM2(6,6),CM3(6,6),
      1DM0,DM1,DM2,V
       V = .00000E+00
       IF(NPASS.GT.1) READ(5,1) V
       IF(V.LT.0.) GO TO 4
       WRITE(6,3)
 3     FORMAT(1H1)
       WRITE(6,2) V
 2     FORMAT(/,10X,*VELOCITY = *,E12.5* FT/SEC*)
 1     FORMAT(E12.5)
       DO 10 I=1,N
       DO 11 J=1,N
       CM0(I,J) = C0(J,I)*CONST*V**4
       CM1(I,J) = C1(J,I)*CONST*B*V**3
       CM2(I,J) = C2(J,I)*CONST*(B*V)**2
 11    CM3(I,J) = C3(J,I)*CONST*V*B**3
 10    CONTINUE
       DM0 = D0*V**2
       DM1 = D1*B*V
       DM2 = D2*B**2
 4     CONTINUE
       RETURN
       END
C*********************************************
       SUBROUTINE MULTMM(A,B,C,N,M,L)
C*********************************************
C THIS ROUTINE PERFORMS MATRIX MULTIPLICATION FOR MATRIX
C (A) WITH DIMENSIONS N,M TIMES MATRIX (B) WITH DIMENSIONS
C M,L AND STORED IN MATRIX (C) WITH DIMENSIONS N,L
       DIMENSION A(4,4),B(4,4),C(4,4),V(4)
       DO 1 I=1,N
       DO 2 J=1,L
       DO 3 K=1,M
 3     TOT = TOT+A(I,K)*B(K,J)
 2     V(J) = TOT
       DO 4 J=1,L
 4     C(I,J) = V(J)
```

```
 1     CONTINUE
       RETURN
       END
C**********************************
       SUBROUTINE INVERT(A,N)
C**********************************
C THIS ROUTINE PERFORMS MATRIX INVERSION (LINPACK ROUTINES).
C INVERT MATRIX (A) WITH DIMENSIONS N,N AND STORE IN
C MATRIX (A)
       DIMENSION A(4,4),V1(4),V2(4),DET(2)
       CALL SGECO(A,4,N,V1,RCOND,V2)
       CALL SGEDI(A,4,N,V1,DET,V2,1)
       RETURN
       END
       RETURN
       END
C*************************************
       SUBROUTINE WRITE4(A,N,M)
C*************************************
       DIMENSION A(4,4)
       WRITE(6,110)
       DO 1 I=1,N
 1     WRITE(6,100) (A(I,J),J=1,M)
 100   FORMAT(1X,4E12.5)
 110   FORMAT(//)
       RETURN
       END
C***************************************
       SUBROUTINE WRITE6(A,N,M)
C***************************************
       DIMENSION A(6,6)
       WRITE(6,110)
       DO 1 I=1,N
 1     WRITE(6,100) (A(I,J),J=1,M)
 100   FORMAT(1X,6E12.5)
 110   FORMAT(//)
       RETURN
       END
C*****************************************************************
       SUBROUTINE PLOT(X,Y,LX,A,IC,B,MQ,LL,XQELT,TITLE,IDM,TAPEN)
C*****************************************************************
C PLOTR, A  POINT PLOT ROUTINE ADAPTED FROM 360 PLOT
C PLOTS A GRAPH OF ONE OR MORE CURVES FROM GIVEN SETS OF
C RECTANGULAR COORDINATES
C
C CALL PLOTR(X,Y,M,A,IC,B,MP,LL,XDELT,TITLE,IDM)
C
C  X      NAME OF A 2 DIMENSIONAL ARRAY CONTAINING THE X COORD-
C         INATES OF ALL THE CURVES TO BE PLOTTED. THE X COORD-
C         INATES OF THE NTH CURVE, FOR EXAMPLE, ARE STORED FROM
C         X(1,N) TO X(M,N) WHERE M IS THE NUMBER OF POINTS IN
C         THE CURVE
```

```
C
C  Y       HAS THE SAME DIMENSIONS AS X AND CONTAINS THE Y
C          COORDINATES.
C
C  M       NAME OF ONE DIMENSIONAL ARRAY SET UP BY THE USER.
C          M(N) IS THE NUMBER OF POINTS IN THE N-TH CURVE, WHOSE
C          FIRST POINT IS AT X(1,N) AND Y(1,N).
C
C  A       NAME OF ONE DIMENSIONAL ARRAY SET UP BY THE USER.
C          A(N) IS THE CHARACTER--LEFT ADJUSTED IN A 4 BYTE WORD
C          TO BE USED IN PLOTTING THE N-TH CURVE.
C
C  IC      THE NUMBER OF CURVES TO BE PLOTTED.  IT MUST BE LESS
C          THAN OR EQUAL TO THE SECOND DIMENSION SPECIFIED FOR
C          THE X AND Y ARRAYS IN THEIR DIMENSION STATEMENT.
C
C  B       1  NO BORDER, NO AXIS
C          2  BORDER, NO AXIS
C          3  NO BORDER, X AXIS ONLY
C          4  BORDER, X AXIS ONLY
C          5  NO BORDER, Y AXIS ONLY
C          6  BORDER, Y AXIS ONLY
C          7  NO BORDER, BOTH AXES
C          8  BORDER, BOTH AXES
C
C  MP      0 SINGLE PAGE PLOT DESIRED
C          1 MULTIPLE PAGE OR FRACTION OF PAGE PLOT DESIRED
C
C  L       1 PLOT GIVEN POINTS
C          2 SEMILOG (PLOT LOGS OF Y COORDINATES)
C          3 LOG-LOG (PLOT LOGS OF X AND Y COORDINATES)
C
C XDELT  0  INDICATES DELTA X IS TO BE CALCULATED
C             OTHERWISE SPECIFY DELTA X IN FLOATING POINT.
C
C  TITLE   NAME OF THE ARRAY IN WHICH THE TITLE TO HEAD EACH
C          PAGE IS STORED.  TEN WORDS ARE ALWAYS PRINTED.
C
C  IDM     THE FIRST DIMENSION SPECIFIED FOR THE X AND Y ARRAYS
C          IN THEIR DIMENSION STATEMENT.  IT MUST BE AT LEAST
C          EQUAL TO THE NUMBER OF POINTS IN THE CURVE WHICH HAS
C          THE GREATEST NUMBER OF POINTS OF THE CURVES TO BE
C          PLOTTED.
C
C
C  TAPEN   OUTPUT TAPE FOR PLOTR
C
C
      DIMENSION X(1),Y(1),LX(1),A(1),PLINE(11),XAX(6),TITLE(12)
      INTEGER A,SS,PLINE,B,BLNK,PLUS,TAPEN
      DATA PLUS,MINUS,II,BLNK /1H+,1H*,1HI,10H
   11 FORMAT(1H+, 5X,12A10 )
```

```
   12 FORMAT(17H   SCALE/INCH    X=,1PE14.7,5H    Y=,1PE14.7,10X,
     ."+OR- TOLERANCE/POINT    X=",1PE14.7,5H    Y=,1PE14.7)
   13 FORMAT(11X,11A10)
   14 FORMAT(1X,E9.2,1X,11A10)
   15 FORMAT(11X,5(1H+,19X),1H+/7X,5(E9.2,11X),E9.2)
   16 FORMAT(17H      MINIMUM    X=,1PE14.7,5H    Y=,1PE14.7,23X,
     ."MAXIMUM   X=",1PE14.7,5H    Y=,1PE14.7)
      XMAX=X(1)
      YMAX=Y(1)
      XDELT=ABS(XQELT)
      MP=MQ
      IF(MP.EQ.1.AND.XDELT.EQ.0.)MP=0
C  INITIALIZATION IS NOW COMPLETE
C  NOW CURVES WILL BE SEARCHED FOR XMAX,XMIN,YMAX,YMIN,AND
C  XDELT=MINIMUM DISTANCE BETWEEN ANY TWO POINTS ON
C  ANY CURVE
      GO TO(3,2,1),LL
    1 XMAX=ABS(XMAX)
    2  YMAX=ABS(YMAX)
    3  XMIN=XMAX
      YMIN=YMAX
      L=2
      IF(XDELT .NE. 0.) GO TO 5
    4  L=1
    5  DO 400 J=1,IC
      N=1+IDM*(J-1)
      XCMAX=X(N)
      YCMAX=Y(N)
      GO TO(22,21,20),LL
   20 XCMAX=ABS(XCMAX)
   21 YCMAX=ABS(YCMAX)
   22 XCMIN=XCMAX
      YCMIN=YCMAX
      GO TO(23,27),L
   23 XCDELT=0
   27 NEL=LX(J)
      DO 300 I=1,NEL
      N=I+IDM*(J-1)
      XTEMP=X(N)
      YTEMP=Y(N)
      GO TO(30,29,28),LL
   28 XTEMP=ABS(XTEMP)
   29 YTEMP=ABS(YTEMP)
   30 XCMAX=AMAX1(XTEMP,XCMAX)
      XCMIN=AMIN1(XTEMP,XCMIN)
      YCMAX=AMAX1(YTEMP,YCMAX)
      YCMIN=AMIN1(YTEMP,YCMIN)
      GO TO(31,300),L
   31 IF(I .GE. NEL) GO TO 300
  311  M=I+1
      DO 100 K=M,NEL
      N=K+IDM*(J-1)
```

```
      GO TO(32,32,33),LL
   32 XDIFF=XTEMP-X(N)
      GO TO 335
   33 XDIFF=XTEMP/ABS(X(N))
      IF(XDIFF .EQ. 0.) GO TO 100
 332  XDIFF=ALOG10(XDIFF)
 335  XDIFF=ABS(XDIFF)
      IF(XDIFF .EQ. 0.) GO TO 100
  336 IF(XCDELT .NE. 0.) GO TO 338
  337 XCDELT=XDIFF
      GO TO 100
  338 XCDELT=AMIN1(XDIFF,XCDELT)
  100 CONTINUE
  300 CONTINUE
      XMAX=AMAX1(XMAX,XCMAX)
      XMIN=AMIN1(XMIN,XCMIN)
      YMAX=AMAX1(YMAX,YCMAX)
      YMIN=AMIN1(YMIN,YCMIN)
      GO TO(34,400),L
 34   IF(XDELT  .NE. 0.) GO TO 346
 345  XDELT=XCDELT
      GO TO 400
 346  XDELT=AMIN1(XDELT,XCDELT)
  400 CONTINUE
C  IF XDELT IS STILL ZERO THEN THE CURVES ARE PARALLEL TO THE
C   Y AXIS SO XMAX,XMIN AND XDELT ARE CALCULATED BY OTHER MEANS
C    DESIGNED TO CENTER CURVES ON A SINGLE PAGE
      IF(XDELT  .NE. 0.) GO TO 365
  347 IF(XMAX)354,353,352
  353 XMAX=10.+XMAX
      XMIN=XMIN-10.
      GO TO 348
  352 XMIN=XMIN-XMAX
      XMAX=2.*XMAX
      GO TO 348
  354 XMAX=XMAX-XMIN
      XMIN=2.*XMIN
      GO TO 348
  365 IF(XMAX .NE. XMIN) GO TO 3365
 3366 XMIN=XMIN-54.*XDELT
      XMAX=XMAX+54.*XDELT
  348 MP=0
C  IF THE CURVES ARE PARALLEL TO THE Y AXIS,A SIMILAR PROCEDURE
C  IS FOLLOWED FOR YMAX,YMIN,ANDYDELT
 3365 IF(YMAX .NE. YMIN) GO TO 366
  349 IF(YMAX)362,363,364
  363 YMAX=10.+YMAX
      YMIN=YMIN-10.
      GO TO 351
  364 YMAX=2.*YMAX
      YMIN=YMIN-YMAX
      GO TO 351
```

```
  362 YMIN=2.*YMIN
      YMAX=YMAX-YMIN
  351 MP=0
  366 GO TO(37,358,35),LL
   35 IF(XMAX .EQ. 0.) GO TO 356
  355 XMAX=ALOG10(XMAX)
  356 IF(XMIN .EQ. 0.) GO TO 358
  357 XMIN=ALOG10(XMIN)
  358 IF(YMAX.EQ.0.) GO TO 359
   36 YMAX=ALOG10(YMAX)
  359 IF(YMIN.EQ.0.) GO TO 37
  361 YMIN=ALOG10(YMIN)
   37 YRANGE=YMAX-YMIN
      XRANGE=XMAX-XMIN
      PRANGE=XDELT*108.
C  IF THE SINGLE PAGE OPTION IS DESIRED BUT XDELT IS TOO LARGE
C   FOR THE RANGE OF X VALUES TO FIT ON ONE PAGE,THEN A NEW XDELT
C   MUST BE FOUND
      IF(MP .NE. 0.) GO TO 376
  374 IF(XRANGE .EQ. PRANGE) GO TO 376
  375 XDELT=XRANGE/108.
      PRANGE=XRANGE
  376 NPAGE=XRANGE/PRANGE+1.
      YDELT=YRANGE/50.
      YSF=YRANGE/8.33
      XSF=PRANGE/10.8
      XTP=XDELT/2.
      YTP=YDELT/2.
      PXMAX=XMIN+XTP
C  NOW THE PLOT IS FORMED A LINE AT A TIME, SEARCHING EACH
C   CURVE ONCE FOR EVERY LINE ON EVERY PAGE AND PRINTING OUT EACH
C  LINE AS SOON AS IT IS FORMED
 39   DO 900 K=1,NPAGE
      IF(PXMAX-XMAX .GE. XTP) GO TO 73
 40   IF(PXMAX .LE. XMAX. GO TO 401
 404  IF(K-1)375,375,73
 401   PXMIN=PXMAX-XDELT
      PXMAX=PXMAX+PRANGE
      IF(K .NE. NPAGE) GO TO 402
 403  PXMAX=XMAX+XTP
 402  GO TO(42,42,41),LL
 41   PNXMN=10.**PXMIN
      PNXMX=10.**PXMAX
   42 CONTINUE
      WRITE(6,6000)
 6000 FORMAT(1H1,//)
      WRITE(6,11) (TITLE(I),I=1,12)
      WRITE(6,6010)
 6010 FORMAT(1H0)
      WRITE(TAPEN,16) XMIN,YMIN,XMAX,YMAX
      WRITE(TAPEN,12) XSF,YSF,XTP,YTP
      GO TO(44,43,43,43,44,43,43,43),9
```

```
43    XTEMP=PXMAX-XTP
      JJJ=(AMIN1(XTEMP,XMAX)-PXMIN)/XDELT
44    YUB=YMAX+YTP
      YLB=YMAX-YTP
      DO 700 LLINE=1,51
      LINE=52-LLINE
      DO 450 I=1,11
450   PLINE(I)=BLNK
      GO TO(46,45,45),LL
45    YNUB=10.**YUB
      YNLB=10.**YLB
46    GO TO(51,47,511,47,519,47,511,47),B
47    N=JJJ/10
      NCHAR=MOD(JJJ,10)
      IF(LINE-1)50,49,48
48    IF(LINE .NE. 51) GO TO 50
49    IF(N .LE. 0) GO TO 549
5549  DO 500 I=1,N
 500  PLINE(I)=10H++++++++++
549   NCHAR=NCHAR+1
      DO 550 JJ=1,NCHAR
      J=JJ-1
 550  CALL PACK(PLINE(N+1),PLUS,J)
      GO TO 5118
  50  CALL PACK(PLINE(1),PLUS,0)
      CALL PACK(PLINE(N+1),PLUS,NCHAR)
5118  GO TO(51,51,511,511,519,519,511,511),B
511   IF(YUB)512,513,514
514   IF(YLB .GE. 0.) GO TO 512
513   N=JJJ/10
      NCHAR=MOD(JJJ,10)+1
      IF(N .LE. 0) GO TO 5515
5513  DO 515 I=1,N
 515  PLINE(I)=10H**********
5515  DO 521 I=1,NCHAR
      J=I-1
 521  CALL PACK(PLINE(N+1),MINUS,J)
512   GO TO (51,51,51,51,519,519,519,519),B
519   IF(PXMAX)51,516,517
517   IF(PXMIN .GT. 0.) GO TO 51
516   I=-PXMIN/XDELT
      NCHAR=MOD(I,10)
      I=I/10+1
      CALL PACK(PLINE(I),II,NCHAR)
51    DO 600 J=1,IC
      NEL=LX(J)
      DO 600 I=1,NEL
      M=I+IDM*(J-1)
      GO TO(54,52,52),LL
52    YTEMP=ABS(Y(M))
      IF(YTEMP-YNLB)600,58,53
53    IF(YTEMP-YNUB)58,600,600
```

```
54    IF(Y(M)-YLB)600,56,55
55    IF(Y(M).GE.YUB) GO TO 600
56    XTEMP=X(M)
      IF(XTEMP-PXMIN)600,62,57
57    IF(XTEMP-PXMAX)62,62,600
58    GO TO(56,56,59),LL
59    XTEMP=ABS(X(M))
      IF(XTEMP-PNXMN)600,61,60
60    IF(XTEMP.GT.PNXMX) GO TO 600
61    IF(XTEMP.EQ.0.) GO TO 62
611   XTEMP=ALOG10(XTEMP)
62    JJ=(XTEMP-PXMIN)/XDELT
      GO TO(66,64,63),LL
63    IF(X(M).LT.0.) GO TO 65
64    IF(Y(M).GE.0.) GO TO 66
65    SS=1HN
      GO TO 67
66    SS=A(J)
67    NCHAR=MOD(JJ,10)
      N=JJ/10+1
      CALL PACK(PLINE(N),SS,NCHAR)
600   CONTINUE
68    IF(MOD(LINE,6).NE.1) GO TO 69
901   IF(LINE .NE. 1)GO TO 71
902   WRITE(TAPEN,14) YMIN,PLINE
      GO TO 715
69    IF(LINE .EQ. 51) GO TO 71
70    WRITE(TAPEN,13) PLINE
      GO TO 715
71    YTEMP=YLB+YTP
      WRITE(TAPEN,14) YTEMP,PLINE
715   YUB=YUB-YDELT
      YLB=YLB-YDELT
700   CONTINUE
72    DO 800 I=1,6
      J=I-1
      XAX(I)=PXMIN+XTP+XDELT*FLOAT(J)*20.
800   CONTINUE
      WRITE(TAPEN,15) XAX
900   CONTINUE
73    RETURN
      END
      SUBROUTINE PACK(WORD,CHAR,IPOS)
      DIMENSION SPLIT(10)
      DATA SPLIT/10*1H /
      DECODE(10,2,WORD) (SPLIT(I),I=1,10)
      SPLIT(IPOS+1)=CHAR
      ENCODE(10,2,WORD) (SPLIT(I),I=1,10)
      RETURN
    2 FORMAT(10A1)
      END
      SUBROUTINE PLOT1
```

```
      DIMENSION S(1),TITLE(12),X(240),Y(240)
      COMMON /PL/ JV,IV,XA(960),YA(960)
      DATA S /1H*/,TITLE /120HROOT LOCUS OF PASSIVE SYSTEM
     1
     2                                                /
      J = 0
      DO 1 I=721,960
      J = J+1
      X(J) = XA(I)
    1 Y(J) = YA(I)
      CALL PLOT(X,Y,240,S,1,7,0,1,0,TITLE,240,6)
      RETURN
      END
      SUBROUTINE PLOT2
      DIMENSION S(1),TITLE(12)
      COMMON /PL/ JV,IV,XA(960),YA(960)
      DATA S /1H*/,TITLE /120HROOT LOCUS OF AUGMENTED SYSTEM
     1
     2                                                /
      CALL PLOT(XA,YA,720,S,1,7,0,1,0,TITLE,720,6)
      RETURN
      END
      SUBROUTINE PLOT3
      DIMENSION S(1),TITLE(12)
      COMMON /PL/ JV,IV,XA(960),YA(960)
      DATA S /1H*/,TITLE /120HCOMBINED ROOT LOCUS
     1
     2                                               /
      CALL PLOT(XA,YA,JV,S,1,7,0,1,0,TITLE,JV,6)
      RETURN
      END
      SUBROUTINE PLOT4(G,W,V,N)
      DIMENSION G(161),W(161),V(240),S(1),TITLE1(12),TITLE2(12)
      DATA S /1H*/,TITLE1 /120HSTRUCTURAL DAMPING VERSUS
     1VELOCITY
     2                                               /
      DATA TITLE2 /120HFREQUENCY VERSUS VELOCITY
     1
     2                                               /
      CALL PLOT(V,G,N,S,1,7,0,1,0,TITLE1,N,6)
      CALL PLOT(V,W,N,S,1,7,0,1,0,TITLE2,N,6)
      RETURN
      END
```

Sample Input File for RLOCUS Program:

```
    3    0    2    0    0
1.          .002378     1.          1000.       1000.
.026832     .025608     .12
0.0
6.0624      161.23      2649.8
 -.28559E+00  .77822E-01 -.10054E+01 -.41821E-01 -.59417E-02 -.12356E+01
 -.81900E-02 -.31297E+01 -.15436E+00 -.68454E-01 -.13846E+02  .33391E+01
 -.46969E+02 -.20996E+01 -.32022E+00 -.20622E+00 -.30516E+00 -.20707E+01
 -.11302E+01  .25951E-01  .34344E+01 -.16266E+01  .43709E+01  .23344E+00
  .94072E+00
  .10335E+01  .94280E-01  .30429E+01  .16542E+00  .13270E-01 -.16317E+01
  .10079E+01 -.49820E+01 -.17166E+00 -.14372E+00 -.15528E+02  .56048E+01
 -.39928E+02 -.15323E+01 -.90763E+00 -.35314E+00 -.27844E+00 -.23707E+01
 -.11759E+01  .17966E-01  .36789E+01 -.17592E+01  .21658E+01  .17140E+00
  .15132E+01
  .12065E+01  .33364E-01  .29650E+01  .15805E+00  .45343E-01 -.60696E+00
  .12636E+01 -.22004E+01 -.38022E-01 -.98991E-01 -.36239E+01  .33487E+01
  .12790E+01  .27846E+00 -.66577E+00 -.17243E+00  .55024E-01 -.35475E+00
 -.58786E-02 -.92806E-02  .75659E+00 -.42457E+00 -.55406E-01  .12865E-01
  .60181E+00
  .28137E+00 -.11552E+00  .34789E+00  .65191E-02  .31736E-01 -.13467E+00
  .32762E+00 -.36052E+00 -.31621E-02 -.20013E-01  .16029E+00 -.19255E+00
  .87869E+00  .70012E-02 -.13154E-01  .77903E-02 -.28798E-02  .13401E-01
  .13624E-02  .70142E-03  .54399E-01 -.26290E-01  .31042E-01  .24164E-02
  .34408E-01
  .10418E+01  .25446E 02
94.
115.
156.
187.
203.
219.
-1.
```

Appendix C: State Model Manipulation Program

Contained on the following pages is a listing of the program STMOD which conducts several state model manipulations at the users' discretion. The input matrices vary depending on the option selected. Currently, there are five selectable options available to the user which are as follows:

1) Singular Value Decomposition and LU Decomposition of the Observability Matrix

2) Singular Value Decomposition and LU Decomposition of the Controllability Matrix

3) State Transformation of a Matrix to Diagonal Form (Eigenvalues and Eigenvectors Calculated)

4) Enter the Feedback Gain Matrix G and Form A + BG Matrix and Find Closed Loop Eigenvalues

5) Do Same as 4 but in Addition Read in Observer Gain Matrix and Find Observer Poles

The state coefficient matrices can be input manually (and are prompted for by the program) or read from a data file attached as TAPE8. To execute this program on the CYBER 750/74, the following commands are necessary:

```
/ATTACH,IMSL4/UN=APPLIB.
/LIBRARY,IMSL4.
/GET,TAPE8=(DATA FILE NAME-NOT NECESSARY FOR MANUAL INPUT)
/GET,LGO=STMOD/UN=E710067.
/LGO.
```

The user is then prompted for the required inputs, and the output is contained in the local file DATA and can be cataloged by the user for further use.

```
      PROGRAM STMOD(INPUT=/80,OUTPUT=/80,DATA=/80,TAPE5=INPUT,
     &TAPE6=OUTPUT,TAPE7=DATA,TAPE8=/80)
C
C     THIS PROGRAM IS USED TO PERFORM SEVERAL MANIPULATIONS OF A
C     SYSTEM IN STATE VARIABLE FORM (USING IMSL4 LIBRARY ROUTINES)
C
C       I.E.        XDOT = AX + BU
C                   Y = CX
C
C
C     WHERE
C
C       A=STATE COEFFICIENT MATRIX
C       B=CONTROL COEFFICIENT MATRIX
C       C=OUTPUT COEFFICIENT MATRIX
C
C
      REAL A(12,12),B(12,12),C(12,12),CA(12,12),Q(12,12),Q1(12,12)
      REAL P(12,12),UT1(12,12),V(12,12),V1(12,12),CA1(12,12)
      REAL WK(24),UT(12,12),S(12),BA(12,12),BA1(12,12),P1(12,12)
      REAL LU(12,12),EQUIL(12),L(12,12),U(12,12),WK3(180)
      REAL AT(12,12),WK1(168),TREAL(12,12),TINVR(12,12)
      REAL ER(12),EI(12),ETR(12),ETI(12),TRID(12,12),TLU(12,12)
      REAL EVMAG(12),WK2(12),ATR(12,12),TINVBR(12,12),TTRAN(12,12)
      REAL G(12,12),BG(12,12),ABG(12,12),WK4(12),BGTI(12,12)
      REAL T9X9(12,12),TT9X9(12,12),TTT9(12,12),AMAT(12,12)
      REAL CTREAL(12,12),K1(12,12),KCT(12,12),AE(12,12),AE1(12,12)
      REAL AE2(12,12),AE2A(12,12),ZREAL(12,12),CTT(12,12)
      REAL K1T(12,12),B9(12,12),A9(12,12),T9(12,12),TTT(12,12)
      REAL T9I(12,12),BG9(12,12),ABG9(12,12),A91(12,12)
      REAL A9CL(12,12),G9(12,12),BMAT(12,12)
      INTEGER IPVT(12)
      COMPLEX EVAL(12),EVALT(12),Z(12,12),TT(12,12),ZN,ZNT
      COMPLEX AC(12,12),T1(12,12),T(12,12),ZTT(12,12)
      COMPLEX TINVC(12,12),WA(168),TID(12,12),ZTEMP(12,12)
      COMPLEX BC(12,12),TINVB(12,12),BZ(12,12)
      COMMON/INOUT/NIN,NOUT,MOUT
C
C     INITIALIZE ARRAYS
C
      DO 1 I=1,12
      IPVT(I)=0.0
      EQUIL(I)=0.0
      S(I)=0.0
      ER(I)=0.0
      EI(I)=0.0
      ETR(I)=0.0
      ETI(I)=0.0
      EVMAG(I)=0.0
      WK2(I)=0.0
      WK4(I)=0.0
      BZ(I)=CMPLX(0.0,0.0)
```

```
EVAL(I)=CMPLX(0.0,0.0)
EVALT(I)=CMPLX(0.0,0.0)
DO 1 J=1,12
A(I,J)=0.0
B(I,J)=0.0
C(I,J)=0.0
CA(I,J)=0.0
Q(I,J)=0.0
P(I,J)=0.0
UT1(I,J)=0.0
V(I,J)=0.0
LT(I,J)=0.0
G1(I,J)=0.0
BA(I,J)=0.0
BA1(I,J)=0.0
P1(I,J)=0.0
LU(I,J)=0.0
L(I,J)=0.0
U(I,J)=0.0
TLU(I,J)=0.0
TREAL(I,J)=0.0
TINVR(I,J)=0.0
TRID(I,J)=0.0
ATR(I,J)=0.0
TINVBR(I,J)=0.0
TTRAN(I,J)=0.0
G(I,J)=0.0
BG(I,J)=0.0
ABG(I,J)=0.0
BGTI(I,J)=0.0
TTT(I,J)=0.0
T9X9(I,J)=0.0
TT9X9(I,J)=0.0
TTT9(I,J)=0.0
AMAT(I,J)=0.0
BMAT(I,J)=0.0
CTREAL(I,J)=0.0
K1(I,J)=0.0
KCT(I,J)=0.0
AE(I,J)=0.0
AE1(I,J)=0.0
AE2(I,J)=0.0
AE2A(I,J)=0.0
ZREAL(I,J)=0.0
CTT(I,J)=0.0
K1T(I,J)=0.0
B9(I,J)=0.0
G9(I,J)=0.0
A9(I,J)=0.0
T9(I,J)=0.0
T9I(I,J)=0.0
BG9(I,J)=0.0
```

```
      ABG9(I,J)=0.0
      A91(I,J)=0.0
      A9CL(I,J)=0.0
      BC(I,J)=CMPLX(0.0,0.0)
      TINVB(I,J)=CMPLX(0.0,0.0)
      ZTEMP(I,J)=CMPLX(0.0,0.0)
      TINVC(I,J)=CMPLX(0.0,0.0)
      TID(I,J)=CMPLX(0.0,0.0)
      Z(I,J)=CMPLX(0.0,0.0)
      TT(I,J)=CMPLX(0.0,0.0)
      ZTT(I,J)=CMPLX(0.0,0.0)
      AC(I,J)=CMPLX(0.0,0.0)
      T1(I,J)=CMPLX(0.0,0.0)
      T(I,J)=CMPLX(0.0,0.0)
 1    CONTINUE
      DO 2 I=1,24
 2    WK(I)=0.0
C  SET UT EQUAL TO IDENTITY MATRIX ON INPUT
      DO 3 J=1,12
 3    UT(J,J)=1.0
      DO 4 I=1,168
      WK1(I)=0.0
      WA(I)=CMPLX(0.0,0.0)
 4    CONTINUE
      NIN=5
      NOUT=6
      MOUT=7
      KIN=8
      JPASS=1
      WRITE(MOUT,310)
 310  FORMAT(1H1)
      WRITE(NOUT,100)
 100  FORMAT(/,1X,"ENTER N, THE ORDER OF THE SYSTEM >")
      READ(NIN,*) NSYS
      WRITE(NOUT,110)
 110  FORMAT(/,1X,"ENTER M, THE ORDER OF THE CONTROL >")
      READ(NIN,*) MC
C***************************
      N=NSYS
      M=MC
C***************************
      WRITE(NOUT,102)
 102  FORMAT(//,1X,"***** MENU - ENTER ONE CHOICE : *****",//,1X,
     &"(1) SINGULAR VALUE DECOMPOSITION AND LU DECOMPOSITION",/,
     &1X,"     OF OBSERVABILITY MATRIX G",//,
     &1X,"(2) SINGULAR VALUE DECOMPOSITION AND LU DECOMPOSITION",
     &/,1X,"     OF CONTROLLIBILITY MATRIX P",//,1X,
     &"(3) STATE TRANSFORMATION OF A MATRIX TO DIAGONAL FORM",/
     &,1X,"     X = TZ  (EIGENVALUES AND EIGENVECTORS CALCULATED)"
     &,//,1X,"(4) ENTER GAIN MATRIX G AND FORM A+BG MATRIX AND"
     &,/,1X,"     FIND CLOSED LOOP EIGENVALUES",//,
     &1X,"(5) DO SAME AS (4) BUT IN ADDITION READ IN",/,
```

```
     &1X,"    OBSERVER GAIN MATRIX AND FIND OBSERVER",
     &" EIGENVALUES",//)
      READ(NIN,*) ICHOICE
      WRITE(NOUT,113)
113   FORMAT(//,1X,"DO YOU WANT TO ENTER STATE MATRICES (ENTER 1)"
     &,/,1X,"OR DO YOU WANT TO READ THEM FROM A FILE (ENTER 2)
     & ?",/)
      READ(NIN,*) KREAD
      IF(KREAD.NE.2) GO TO 9
      REWIND KIN
      CALL READF(A,N,N,KIN)
      WRITE(MOUT,205)
      CALL WRITE(A,N,N,MOUT)
      IF(ICHOICE.EQ.1) GO TO 8
      CALL READF(B,N,M,KIN)
      WRITE(MOUT,210)
      CALL WRITE(B,N,M,MOUT)
      IF(ICHOICE.LT.4) GO TO 13
      CALL READF(G,M,N,KIN)
      WRITE(MOUT,212)
212   FORMAT(//,1X,"FEEDBACK GAIN MATRIX G :",/)
      CALL WRITE(G,M,N,MOUT)
      IF(ICHOICE.LT.5) GO TO 13
      CALL READF(K1,N,M,KIN)
      WRITE(MOUT,213)
213   FORMAT(//,1X,"OBSERVER GAIN MATRIX K :",/)
      CALL WRITE(K1,N,M,MOUT)
      GO TO 13
8     CONTINUE
      CALL READF(C,M,N,KIN)
      WRITE(MOUT,215)
      CALL WRITE(C,M,N,MOUT)
      GO TO 13
9     CONTINUE
      WRITE(NOUT,115)
115   FORMAT(//,1X,"USE LIST DIRECTED READ FOR THE ARRAYS",/,1X,
     &"ENTER I,J,NON-ZERO ENTRY A(I,J) ",//)
10    WRITE(NOUT,120)
120   FORMAT(/,1X,"ENTER A MATRIX (N X N) :",/)
      CALL READ(A,N,N,NIN)
      CALL WRITE(A,N,N,NOUT)
      WRITE(NOUT,200)
200   FORMAT(1X,"ENTER 0 TO ACCEPT, 1 TO CHANGE >")
      READ(NIN,*) IANS
      IF(IANS.EQ.1) GO TO 10
      WRITE(MOUT,205)
205   FORMAT(/,1X,"A MATRIX :",/)
      CALL WRITE(A,N,N,MOUT)
      IF(ICHOICE.EQ.1) GO TO 40
12    WRITE(NOUT,125)
125   FORMAT(/,1X,"ENTER SYSTEM B MATRIX (N X M) :",/)
      CALL READ(B,N,M,NIN)
```

```
      CALL WRITE(B,N,M,NOUT)
      WRITE(NOUT,200)
      READ(NIN,*) IANS
      IF(IANS.EQ.1) GO TO 12
      WRITE(MOUT,210)
210   FORMAT(/,1X,"SYSTEM B MATRIX  :",/)
      CALL WRITE(B,N,M,MOUT)
      IF(ICHOICE.LT.4) GO TO 15
11    WRITE(NOUT,127)
127   FORMAT(/,1X,"ENTER FEEDBACK GAIN MATRIX G (MXN):",/)
      CALL READ(G,M,N,NIN)
      CALL WRITE(G,M,N,NOUT)
      WRITE(NOUT,200)
      READ(NIN,*) IANS
      IF(IANS.EQ.1) GO TO 11
      WRITE(MOUT,212)
      CALL WRITE(G,M,N,MOUT)
      IF(ICHOICE.LT.5) GO TO 13
15    WRITE(NOUT,129)
129   FORMAT(/,1X,"ENTER OBSERVER GAIN MATRIX K (NXM):",/)
      CALL READ(K1,N,M,NIN)
      CALL WRITE(K1,N,M,NOUT)
      WRITE(NOUT,200)
      READ(NIN,*) IANS
      IF(IANS.EQ.1) GO TO 15
      WRITE(MOUT,213)
      CALL WRITE(K1,N,M,MOUT)
      GO TO 13
40    CONTINUE
14    WRITE(NOUT,130)
130   FORMAT(/,1X,"ENTER SYSTEM C MATRIX (M X N) :",/)
      CALL READ(C,M,N,NIN)
      CALL WRITE(C,M,N,NOUT)
      WRITE(NOUT,200)
      READ(NIN,*) IANS
      IF(IANS.EQ.1) GO TO 14
      WRITE(MOUT,215)
215   FORMAT(/,1X,"SYSTEM C MATRIX :",/)
      CALL WRITE(C,M,N,MOUT)
13    CONTINUE
      CALL EQUATE(A,AMAT,N,N)
      CALL EQUATE(B,BMAT,N,M)
      DO 5 I=1,N
      DO 5 J=1,N
      AC(I,J)=CMPLX(A(I,J),0.0)
5     CONTINUE
C     CALL WRITC(AC,N,N,MOUT)
      CALL MTRAN(A,N,N,AT)
C     CALL WRITE(AT,N,N,MOUT)
      DO 6 I=1,N
      DO 6 J=1,M
      BC(I,J)=CMPLX(B(I,J),0.0)
```

```
 6     CONTINUE
C      CALL WRITC(BC,N,M,MOUT)
       IF(ICHOICE.EQ.2) GO TO 50
       IF(ICHOICE.GE.3) GO TO 41
C*****************************************
C      OBSERVABILITY MATRIX SECTION
C*****************************************
       N=NSYS
       M=MC
       CALL EQUATE(C,CA,M,N)
       CALL WRITE(CA,M,N,MOUT)
       CALL WRITE(CA,M,N,NOUT)
       IT1=0
       NN=N/2
       DO 70 IT=1,NN
C
       DO 72 I=1,M
       IT1=IT1+1
       DO 72 J=1,N
       G(IT1,J)=CA(I,J)
 72    CONTINUE
C
       CALL MMUL(CA,A,M,N,N,CA1)
       CALL EQUATE(CA1,CA,M,N)
       WRITE(MOUT,220) IT
 220   FORMAT(//,1X,I2,"TH CA :",/)
       CALL WRITE(CA,M,N,MOUT)
 70    CONTINUE
       WRITE(MOUT,310)
       WRITE(MOUT,225)
 225   FORMAT(//,1X,"OBSERVABILITY MATRIX, Q :",/)
       CALL WRITE(Q,N,N,MOUT)
       CALL EQUATE(Q,Q1,N,N)
C
       IA=12
       IU=12
       NU=N
C
       CALL LSVDF(Q,IA,N,N,UT,IU,NU,S,WK,IER)
C
       WRITE(MOUT,400) IER
C
       DO 30 I=1,N
       DO 30 J=1,N
       IF(ABS(Q(I,J)).LT.1E-05) Q(I,J)=0.0
 30    V(I,J)=Q(I,J)
       DO 31 I=1,N
       DO 31 J=1,N
       IF(ABS(UT(I,J)).LT.1E-05) UT(I,J)=0.0
 31    UT1(I,J)=UT(I,J)
       WRITE(MOUT,310)
       WRITE(MOUT,230)
```

```
 230   FORMAT(//,1X,"SINGULAR VALUES OF Q :",/)
       WRITE(MOUT,231) (S(K),K=1,N)
 231   FORMAT(1X,6E12.5,/,1X,6E12.5)
       WRITE(MOUT,235)
 235   FORMAT(//,1X,"UTRANSPOSE MATRIX :",/)
       CALL WRITE(UT1,N,N,MOUT)
       CALL MTRAN(UT1,N,N,UT)
       WRITE(MOUT,236)
 236   FORMAT(//,1X,"U MATRIX :",/)
       CALL WRITE(UT,N,N,MOUT)
       WRITE(MOUT,310)
       WRITE(MOUT,240)
 240   FORMAT(//,1X,"V MATRIX :",/)
       CALL WRITE(V,N,N,MOUT)
       CALL EQUATE(Q1,P,N,N)
C
       GO TO 20
 50    CONTINUE
C*******************************************
C      CONTROLLABILITY MATRIX SECTION
C*******************************************
       N=NSYS
       M=MC
       CALL EQUATE(B,BA,N,M)
       CALL WRITE(BA,N,M,MOUT)
       JT2=0
       NN=N/2
       DO 60 IT=1,NN
C
       WRITE(MOUT,315) IT
 315   FORMAT(//,1X,I2,"TH BA :",/)
       CALL WRITE(BA,N,M,MOUT)
C
       DO 61 I=1,N
       JT1=0
       DO 61 J=1,M
       JT1=J+JT2
       P(I,JT1)=BA(I,J)
 61    CONTINUE
C
       CALL MMUL(A,BA,N,N,M,BA1)
       CALL EQUATE(BA1,BA,N,M)
       JT2=JT2+2
 60    CONTINUE
       WRITE(MOUT,310)
       WRITE(MOUT,320)
 320   FORMAT(//,1X,"CONTROLLABILITY MATRIX, P :",/)
       CALL WRITE(P,N,N,MOUT)
       CALL EQUATE(P,P1,N,N)
C
       IA=12
       IU=12
```

```
      NU=N
C
C     CALLING SINGULAR VALUE DECOMPOSITION ROUTINE
C
      CALL LSVDF(P,IA,N,N,UT,IU,NU,S,WK,IER)
C
      WRITE(MOUT,400) IER
      DO 65 I=1,N
      DO 65 J=1,N
      IF(ABS(P(I,J)).LT.1E-05) P(I,J)=0.0
 65   V(I,J)=P(I,J)
      DO 66 I=1,N
      DO 66 J=1,N
      IF(ABS(UT(I,J)).LT.1E-05) UT(I,J)=0.0
 66   UT1(I,J)=UT(I,J)
C
      WRITE(MOUT,310)
      WRITE(MOUT,330)
 330  FORMAT(//,1X,"SINGULAR VALUES OF P :",/)
      WRITE(MOUT,331) (S(K),K=1,N)
 331  FORMAT(1X,6E12.5,/,1X,6E12.5)
      WRITE(MOUT,332)
 332  FORMAT(//,1X,"U TRANSPOSE MATRIX :",/)
      CALL WRITE(UT1,N,N,MOUT)
      CALL MTRAN(UT1,N,N,UT)
      WRITE(MOUT,333)
 333  FORMAT(//,1X,"U MATRIX :",/)
      CALL WRITE(UT,N,N,MOUT)
      WRITE(MOUT,310)
      WRITE(MOUT,340)
 340  FORMAT(//,1X,"V MATRIX :",/)
      CALL WRITE(V,N,N,MOUT)
C
      CALL EQUATE(P1,P,N,N)
 20   CONTINUE
      WRITE(MOUT,310)
      IDGT=3
C
C     CALLING LU DECOMPOSITION ROUTINE
C
      CALL LUDATF(P,LU,N,IA,IDGT,D1,D2,IPVT,EQUIL,WA,IER)
C
      WRITE(MOUT,400) IER
      DET=(D1*(2**D2))
C
      DO 73 I=1,N
      DO 73 J=1,N
      IF(J.GT.I) GO TO 71
      IF(J.EQ.I) GO TO 74
      L(I,J)=LU(I,J)
      GO TO 73
 71   CONTINUE
```

```
      L(I,J)=0.0
      GO TC 73
 74   CONTINUE
      L(I,J)=1.0
 73   CONTINUE
C
      DO 80 I=1,N
      DO 80 J=1,N
      IF(J.LT.I) GO TO 81
      U(I,J)=LU(I,J)
      GO TO 80
 81   CONTINUE
      U(I,J)=0.0
 80   CONTINUE
C
      WRITE(MOUT,350)
 350  FORMAT(//,1X,"LU MATRIX :",/)
      CALL WRITE(LU,N,N,MOUT)
      WRITE(MOUT,310)
      WRITE(MOUT,352)
 352  FORMAT(//,1X,"L MATRIX :",/)
      CALL WRITE(L,N,N,MOUT)
      WRITE(MOUT,310)
      WRITE(MOUT,354)
 354  FORMAT(//,1X,"U MATRIX :",/)
      CALL WRITE(U,N,N,MOUT)
      WRITE(MOUT,360) D1,D2,DET
 360  FORMAT(//,1X,"D1 =",E12.5,2X,"D2 =",E12.5,2X,"DET =",
     &E12.5)
      WRITE(MOUT,362)
 362  FORMAT(//,1X,"IPVT VECTOR :",/)
      CALL WRITE(IPVT,N,1,MOUT)
      WRITE(MOUT,364)
 364  FORMAT(//,1X,"EQUIL VECTOR :",/)
      CALL WRITE(EQUIL,N,1,MOUT)
C
C****************************
 41   CONTINUE
C****************************
C
C***************************************************
C     STATE TRANSFORMATION SECTION (X = TZ)
C***************************************************
C
C     CALCULATE EIGENVALUES AND RT EIGENVECTORS OF A
C
      N=NSYS
      M=MC
      IJOB=2
      IA=12
      IT=12
C
```

```
      CALL EIGRF(A,N,IA,IJOB,EVAL,T,IT,WK1,IER)
C
      DO 85 I=1,N
      ER(I)=REAL(EVAL(I))
      IF(ABS(ER(I)).LT.1E-06) ER(I)=0.0
      EI(I)=AIMAG(EVAL(I))
      IF(ABS(EI(I)).LT.1E-06) EI(I)=0.0
      EVAL(I)=CMPLX(ER(I),EI(I))
      EVMAG(I)=(ER(I)**2 + EI(I)**2)**.5
 85   CONTINUE
      WRITE(MOUT,310)
C
      WRITE(MOUT,400) IER
 400  FORMAT(/,1X,"IER =",I3)
      WRITE(MOUT,402)
 402  FORMAT(//,1X,"EIGENVALUES :",//,6X,"REAL",8X,"IMAG",/)
      DO 83 I=1,N
      WRITE(MOUT,404) EVAL(I)
 83   CONTINUE
 404  FORMAT(1X,2E12.5)
      WRITE(MOUT,406)
 406  FORMAT(//,1X,"EIGENVECTORS (COMPLEX TRANSFORMATION MATRIX)"
     &" :",//,5X,3("REAL",8X,"IMAG",8X),/)
      CALL CLEANC(T,N,N)
      CALL WRITC(T,N,N,MOUT)
      WRITE(MOUT,411) WK1(1)
 411  FORMAT(/,1X,"PERFORMANCE INDEX =",E12.5)
C
C     FORM REAL TRANSFORMATION MATRIX
C
      CALL FORM(T,N,N,TREAL)
      CALL EQUATE(TREAL,TLU,N,N)
      CALL MTRAN(TREAL,N,N,TTRAN)
      WRITE(MOUT,408)
 408  FORMAT(//,1X,"TRANSFORMATION MATRIX T (REAL) :",/)
      CALL WRITE(TREAL,N,N,MOUT)
      WRITE(MOUT,310)
      CALL MMUL(TTRAN,TREAL,N,N,N,TTT)
      CALL CLEANR(TTT,N,N)
      WRITE(MOUT,407)
 407  FORMAT(//,1X,"TTRAN*T MATRIX :",/)
      CALL WRITE(TTT,N,N,MOUT)
      WRITE(MOUT,310)
C
C     FORM 9X9 SUB-TRANSFORMATION MATRIX
C
      I1=0
      DO 35 I=1,N
      IF(I.LE.3) GO TO 35
      I1=I1+1
      J1=0
```

```
      DO 36 J=1,N
      IF(J.LE.3) GO TO 36
      J1=J1+1
      T9X9(I1,J1)=TREAL(I,J)
   36 CONTINUE
   35 CONTINUE
      N9=I1
      CALL WRITE(T9X9,N9,N9,MOUT)
      CALL EQUATE(T9X9,T9,N9,N9)
      IA=12
      IDGT=0
C
      CALL LINV2F(T9,N9,IA,T9I,IDGT,WK3,IER)
C
      WRITE(MOUT,419)
  419 FORMAT(//,1X,"T(9X9) INVERSE :",/)
      CALL WRITE(T9I,N9,N9,MOUT)
      CALL MMUL(T9I,T9X9,N9,N9,N9,TTT9)
      CALL WRITE(TTT9,N9,N9,MOUT)
      CALL MTRAN(T9X9,N9,N9,TT9X9)
      CALL MMUL(TT9X9,T9X9,N9,N9,N9,TTT9)
      WRITE(MOUT,310)
      WRITE(MOUT,409)
  409 FORMAT(//,1X,"TTRAN (9X9) * T(9X9) MATRIX :",/)
      CALL CLEANR(TTT9,N9,N9)
      CALL WRITE(TTT9,N9,N9,MOUT)
C
C     CALCULATE INVERSE OF THE REAL TRANSFORMATION MATRIX T
C
      IA=12
      IDGT=0
C
      CALL LINV2F(TLU,N,IA,TINVR,IDGT,WK3,IER)
C
      WRITE(MOUT,400) IER
C
C     CALCULATE THE INVERSE OF THE COMPLEX TRANSFORMATION
C     MATRIX T (A COLUMN AT A TIME)
C
      M=N
      IJOB=1
C
      CALL LEQ2C(T,N,12,TINVC,M,12,IJOB,WA,WK2,IER)
      WRITE(MOUT,400) IER
C
      M=1
      IJOB=2
      DO 87 J=1,N
      DO 88 I=1,N
      IF(I.EQ.J) GO TO 89
      BZ(I)=CMPLX(0.0,0.0)
      GO TO 88
```

```
 85    CONTINUE
       BZ(I)=CMPLX(1.0,0.0)
 86    CONTINUE
       CALL LEQ2C(T,N,12,BZ,M,12,IJOB,WA,WK2,IER)
       WRITE(MOUT,400) IER
       DO 90 I=1,N
       TINVC(I,J)=BZ(I)
 90    CONTINUE
 87    CONTINUE
       WRITE(MOUT,412)
 412   FORMAT(//,1X,"T INVERSE MATRIX (COMPLEX) :",/)
       CALL CLEANC(TINVC,N,N)
       CALL WRITC(TINVC,N,N,MOUT)
       WRITE(MOUT,310)
       WRITE(MOUT,414)
 414   FORMAT(//,1X,"T INVERSE MATRIX (REAL) :",/)
       CALL CLEANR(TINVR,N,N)
       CALL WRITE(TINVR,N,N,MOUT)
       IF(JPASS.EQ.1) GO TO 99
C
C      CALCULATE LEFT EIGENVECTORS OF A
C      ***** RIGHT NOW THIS IS PASSED OVER, BUT IT WORKS *****
C
       WRITE(MOUT,310)
       IJOB=2
       IT=12
C
       CALL EIGRF(AT,N,12,IJOB,EVALT,TT,IT,WK1,IER)
C
       DO 93 I=1,N
       ETR(I)=REAL(EVALT(I))
       IF(ABS(ETR(I)).LT.1E-06) ETR(I)=0.0
       ETI(I)=AIMAG(EVALT(I))
       IF(ABS(ETI(I)).LT.1E-06) ETI(I)=0.0
       EVALT(I)=CMPLX(ETR(I),ETI(I))
 93    CONTINUE
C
       WRITE(MOUT,400) IER
       WRITE(MOUT,402)
       DO 91 I=1,N
       WRITE(MOUT,404) EVALT(I)
 91    CONTINUE
       WRITE(MOUT,406)
       CALL CLEANC(TT,N,N)
       CALL WRITC(TT,N,N,MOUT)
       WRITE(MOUT,411) WK1(1)
 99    CONTINUE
C
       WRITE(MOUT,310)
       CALL MMULC(TINVC,T,N,N,N,TID)
       WRITE(MOUT,430)
 430   FORMAT(//,1X,"TINV*T MATRIX (COMPLEX) :",/)
```

```
      CALL CLEANC(TID,N,N)
      CALL WRITC(TID,N,N,MOUT)
      CALL MMULC(AC,T,N,N,N,T1)
      CALL CLEANC(T1,N,N)
C     CALL WRITC(T1,N,N,MOUT)
      CALL MMULC(TINVC,T1,N,N,N,Z)
      CALL CLEANC(Z,N,N)
C
      WRITE(MOUT,310)
      WRITE(MOUT,420)
 420  FORMAT(//,1X,"DIAGONALIZED A MATRIX (COMPLEX) :",//,5X,
     &3("REAL",8X,"IMAG",8X),/)
      CALL WRITC(Z,N,N,MOUT)
C
      M=MC
      CALL MMULC(TINVC,BC,N,N,M,TINVB)
      WRITE(MOUT,310)
      WRITE(MOUT,422)
 422  FORMAT(//,1X,"TINV*B MATRIX (COMPLEX) :",/)
      CALL WRITC(TINVB,N,M,MOUT)
C
      WRITE(MOUT,310)
      CALL MMUL(TINVR,TREAL,N,N,N,TRID)
      WRITE(MOUT,432)
 432  FORMAT(//,1X,"TINVR*TREAL MATRIX :",/)
      CALL CLEANR(TRID,N,N)
      CALL WRITE(TRID,N,N,MOUT)
      WRITE(MOUT,310)
      CALL MMUL(A,TREAL,N,N,N,ATR)
      CALL CLEANR(ATR,N,N)
C     CALL WRITE(ATR,N,N,MOUT)
      CALL MMUL(TINVR,ATR,N,N,N,ZREAL)
      CALL CLEANS(ZREAL,N,N)
      WRITE(MOUT,424)
 424  FORMAT(//,1X,"BLOCK DIAGONALIZED A MATRIX (REAL) :",/)
      CALL WRITE(ZREAL,N,N,MOUT)
      CALL MMUL(TINVR,B,N,N,MC,TINVBR)
      WRITE(MOUT,426)
 426  FORMAT(//,1X,"TINV*B MATRIX (REAL) :",/)
      CALL WRITE(TINVBR,N,MC,MOUT)
C*********************************************
C     CONTROLLER CLOSED LOOP CALCULATIONS
C*********************************************
      IF(ICHOICE.LT.4) GO TO 52
C
C     FORM 9X9 SUB-SYSTEM COMPONENT MATRICES FROM
C     FULL SYSTEM MATRICES
C
      I1=0
      DO 23 I=1,N
      IF(I.LE.3) GO TO 23
      I1=I1+1
```

```
      DO 24 J=1,M
      B9(I1,J)=TINVBR(I,J)
 24   CONTINUE
 23   CONTINUE
C
      DO 26 I=1,M
      J1=0
      DO 27 J=1,N
      IF(J.LE.3) GO TO 27
      J1=J1+1
      G9(I,J1)=G(I,J)
 27   CONTINUE
 26   CONTINUE
C
      I1=0
      DO 28 I=1,N
      IF(I.LE.3) GO TO 28
      I1=I1+1
      J1=0
      DO 29 J=1,N
      IF(J.LE.3) GO TO 29
      J1=J1+1
      A9(I1,J1)=ZREAL(I,J)
 29   CONTINUE
 28   CONTINUE
C
C     PERFORM FUL 12X12 CLOSED LOOP SYSTEM ANALYSIS
C
      CALL MMUL(TINVBR,G,N,M,N,BG)
      WRITE(MOUT,310)
      CALL WRITE(BG,N,N,MOUT)
      CALL MADD(ZREAL,BG,N,N,1.0,ABG)
      WRITE(MOUT,310)
      WRITE(MOUT,440)
 440  FORMAT(//,1X,"DIAG(A) + BG MATRIX :",/)
      CALL CLEANR(ABG,N,N)
      CALL WRITE(ABG,N,N,MOUT)
C
      IJOB=0
      IABG=12
      IT=12
C
      CALL EIGRF(ABG,N,IABG,IJOB,EVALT,TT,IT,WK4,IER)
C
      DO 54 I=1,N
      ETR(I)=REAL(EVALT(I))
      IF(ABS(ETR(I)).LT.1E-06) ETR(I)=0.0
      ETI(I)=AIMAG(EVALT(I))
      IF(ABS(ETI(I)).LT.1E-06) ETI(I)=0.0
      EVALT(I)=CMPLX(ETR(I),ETI(I))
 54   CONTINUE
      WRITE(MOUT,310)
```

```
      WRITE(MOUT,400) IER
      WRITE(MOUT,402)
      DO 55 I=1,N
      WRITE(MOUT,404) EVALT(I)
 55   CONTINUE
C
C     PLUGGING CONTROL BACK INTO ORIGINAL NON-DIMENSIONALIZED
C     SYSTEM
C
      WRITE(MOUT,310)
      CALL EQUATE(AMAT,A,N,N)
      CALL EQUATE(BMAT,B,N,M)
      CALL WRITE(A,N,N,MOUT)
      CALL WRITE(B,N,M,MOUT)
      WRITE(MOUT,310)
      CALL MMUL(B,G,N,M,N,BG)
      CALL WRITE(BG,N,N,MOUT)
      CALL MMUL(BG,TINVR,N,N,N,BGTI)
      CALL WRITE(BGTI,N,N,MOUT)
      CALL MADD(A,BGTI,N,N,1.0,ABG)
      WRITE(MOUT,310)
      WRITE(MOUT,442)
 442  FORMAT(//,1X,"A + B*G*TINV MATRIX :",/)
      CALL CLEANR(ABG,N,N)
      CALL WRITE(ABG,N,N,MOUT)
C
      IJOB=0
      IABG=12
      IT=12
C
      CALL EIGRF(ABG,N,IABG,IJOB,EVALT,TT,IT,WK4,IER)
C
      DO 56 I=1,N
      ETR(I)=REAL(EVALT(I))
      IF(ABS(ETR(I)).LT.1E-06) ETR(I)=0.0
      ETI(I)=AIMAG(EVALT(I))
      IF(ABS(ETI(I)).LT.1E-06) ETI(I)=0.0
      EVALT(I)=CMPLX(ETR(I),ETI(I))
 56   CONTINUE
      WRITE(MOUT,310)
      WRITE(MOUT,400) IER
      WRITE(MOUT,402)
      DO 57 I=1,N
      WRITE(MOUT,404) EVALT(I)
 57   CONTINUE
C
C     PERFORM 9X9 CLOSED LOOP SYSTEM ANALYSIS
C
      WRITE(MOUT,310)
      CALL WRITE(A9,N9,N9,MOUT)
      CALL WRITE(B9,N9,M,MOUT)
      CALL WRITE(G9,M,N9,MOUT)
```

```
      CALL MMUL(B9,G9,N9,N9,N9,BG9)
      CALL WRITE(BG9,N9,N9,MOUT)
      CALL MADD(A9,BG9,N9,N9,1.0,ABG9)
      WRITE(MOUT,460)
 460  FORMAT(//,1X,"DIAG(A9) + BG9 MATRIX :",/)
      CALL CLEANR(ABG9,N9,N9)
      CALL WRITE(ABG9,N9,N9,MOUT)
C
      IJOB=0
      IABG=12
      IT=12
C
      CALL EIGRF(ABG9,N9,IABG,IJOB,EVALT,TT,IT,WK4,IER)
C
      WRITE(MOUT,400) IER
      WRITE(MOUT,402)
      DO 62 I=1,N9
      WRITE(MOUT,404) EVALT(I)
 62   CONTINUE
      CALL WRITE(T9X9,N9,N9,MOUT)
      CALL WRITE(T9I,N9,N9,MOUT)
      CALL MMUL(ABG9,T9X9,N9,N9,N9,A91)
      CALL MMUL(T9I,A91,N9,N9,N9,A9CL)
      WRITE(MOUT,462)
 462  FORMAT(//,1X,"ORIGINAL SYSTEM CLOSED LOOP A9 MATRIX:",/)
      CALL CLEANR(A9CL,N9,N9)
      CALL WRITE(A9CL,N9,N9,MOUT)
C
      IJOB=0
      IABG=12
      IT=12
C
      CALL EIGRF(A9CL,N9,IABG,IJOB,EVALT,TT,IT,WK4,IER)
C
      WRITE(MOUT,400) IER
      WRITE(MOUT,402)
      DO 63 I=1,N9
      WRITE(MOUT,404) EVALT(I)
 63   CONTINUE
C
C
      C(1,1)=-.001829
      C(1,2)=.03043
      C(1,3)=.6084
C
      C(2,1)=.70683
      C(2,2)=.08840
      C(2,3)=.68922
      WRITE(MOUT,310)
      CALL WRITE(C,M,N,MOUT)
C
      CALL MMUL(C,TREAL,M,N,N,CTREAL)
```

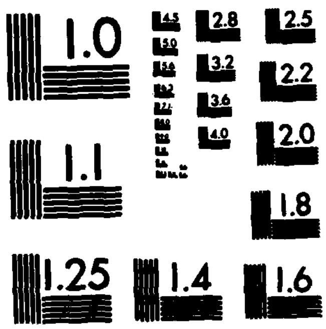
```
      WRITE(MOUT,444)
 444  FORMAT(//,1X,"C*TREAL MATRIX :",/)
      CALL WRITE(CTREAL,M,N,MOUT)
      IF(ICHOICE.LT.5) GO TO 52
C**********************************************
C     OBSERVER CLOSED LOOP CALCULATIONS
C**********************************************
      WRITE(MOUT,310)
      CALL MTRAN(CTREAL,M,N,CTT)
      CALL MTRAN(K1,N,M,K1T)
C     CALL WRITE(CTT,N,M,MOUT)
C     CALL WRITE(K1T,M,N,MOUT)
      CALL MMUL(CTT,K1T,N,M,N,KCT)
C     CALL MMUL(K1,CTREAL,N,M,N,KCT)
      CALL CLEANR(KCT,N,N)
      CALL WRITE(KCT,N,N,MOUT)
      CALL MADD(ZREAL,KCT,N,N,1.0,AE2)
      WRITE(MOUT,448)
 448  FORMAT(//,1X,"OBSERVER (LAMDA - CT*KT) MATRIX :",/)
      CALL WRITE(AE2,N,N,MOUT)
        ALL EQUATE(AE2,AE2A,N,N)
3
      IJOB=0
      IAE=12
        T=12
C
       CALL EIGRF(AE2,N,IAE,IJOB,EVALT,TT,IT,WK4,IER)
C
      WRITE(MOUT,402)
      DO 25 I=1,N
      WRITE(MOUT,404) EVALT(I)
 25   CONTINUE
      CALL EQUATE(AE2A,AE2,N,N)
C     CALL WRITE(AE2,N,N,MOUT)
      CALL MMUL(AE2,TINVR,N,N,N,AE1)
      CALL MMUL(TREAL,AE1,N,N,N,AE)
      WRITE(MOUT,310)
      WRITE(MOUT,450)
 450  FORMAT(//,1X,"CLOSED LOOP ERROR STATE COEFFICIENT MATRIX"
     &" :",/)
      CALL CLEANR(AE,N,N)
      CALL WRITE(AE,N,N,MOUT)
C
      IJOB=0
      IAE=12
      IT=12
C
      CALL EIGRF(AE,N,IAE,IJOB,EVALT,TT,IT,WK4,IER)
C
      DO 21 I=1,N
      ETR(I)=REAL(EVALT(I))
      IF(ABS(ETR(I)).LT.1E-06) ETR(I)=0.0
```

```
      ETI(I)=AIMAG(EVALT(I))
      IF(ABS(ETI(I)).LT.1E-06) ETI(I)=0.0
      EVALT(I)=CMPLX(ETR(I),ETI(I))
 21   CONTINUE
      WRITE(MOUT,400) IER
      WRITE(MOUT,402)
      DO 22 I=1,N
      WRITE(MOUT,404) EVALT(I)
 22   CONTINUE
C
C
 52   CONTINUE
      STOP
      END
C*******************************************
      SUBROUTINE READF(MAT,N,M,LIN)
C*******************************************
C
C     THIS SUBROUTINE READS THE REAL STATE MATRICES
C     FROM DEVICE NUMBER LIN
C
      REAL MAT(12,12)
C
      DO 1 L=1,M,6
      K=L+5
      IF(M-L.LT.5) K=M
      DO 2 I=1,N
      READ(LIN,100) (MAT(I,J),J=L,K)
 2    CONTINUE
 100  FORMAT(6E12.5)
      READ(LIN,110) XXX,YYY
 110  FORMAT(A1,/,A1)
 1    CONTINUE
C
      RETURN
      END
C*******************************************
      SUBROUTINE CLEANR(MAT,N,M)
C*******************************************
C
C     THIS SUBROUTINE ZEROES OUT SMALL ELEMENTS OF
C     A REAL MATRIX MAT(N,M)
C
      REAL MAT(12,12)
C
      DO 1 I=1,N
      DO 1 J=1,M
      IF(ABS(MAT(I,J)).LT.1E-06) MAT(I,J)=0.0
 1    CONTINUE
C
      RETURN
      END
```

```
C*******************************************
      SUBROUTINE CLEANS(MAT,N,M)
C*******************************************
C
      REAL MAT(12,12)
C
      DO 1 I=1,N
      DO 1 J=1,M
      IF(ABS(MAT(I,J)).LT.1E-03) MAT(I,J)=0.0
 1    CONTINUE
C
      RETURN
      END
C*********************************************
      SUBROUTINE FORM(A,N,M,B)
C*********************************************
C
C     THIS SUBROUTINE FORMS A REAL TRANSFORMATION MATRIX
C     FROM A COMPLEX TRANSFORMATION MATRIX
C
      REAL AR(12,12),AI(12,12),B(12,12)
      COMPLEX A(12,12)
C
      DO 1 I=1,N
      DO 1 J=1,M
      AR(I,J)=REAL(A(I,J))
      AI(I,J)=AIMAG(A(I,J))
 1    CONTINUE
      J1=1
      DO 3 J=1,N
      DO 2 I=1,N
      IF(ABS(AI(I,J1)).LT.1E-10) GO TO 2
      GO TO 5
 2    CONTINUE
      DO 8 I=1,N
      B(I,J1)=AR(I,J1)
 8    CONTINUE
      IF(J1.GE.N) GO TO 20
      J1=J1+1
      GO TO 3
 5    CONTINUE
      DO 10 I=1,N
      B(I,J1)=AR(I,J1)
      B(I,J1+1)=AI(I,J1)
 10   CONTINUE
      IF(J1.GE.N) GO TO 20
      J1=J1+2
 3    CONTINUE
 20   CONTINUE
C
      RETURN
      END
```

```
C**********************************************************
      SUBROUTINE CLEANC(MAT,N,M)
C**********************************************************
C
C     THIS SUBROUTINE ZEROES OUT SMALL ELEMENTS OF
C     A COMPLEX MATRIX MAT(N,M)
C
      REAL MATR(12,12),MATI(12,12)
      COMPLEX MAT(12,12)
C
      DO 1 I=1,N
      DO 1 J=1,M
      MATR(I,J)=REAL(MAT(I,J))
      IF(ABS(MATR(I,J)).LT.1E-06) MATR(I,J)=0.0
      MATI(I,J)=AIMAG(MAT(I,J))
      IF(ABS(MATI(I,J)).LT.1E-06) MATI(I,J)=0.0
      MAT(I,J)=CMPLX(MATR(I,J),MATI(I,J))
 1    CONTINUE
C
      RETURN
      END
C*******************************************
      SUBROUTINE EQUATE(A,B,N1,M1)
C*******************************************
C
C     THIS SUBROUTINE EQUATES REAL MATRIX B TO REAL
C     MATRIX B (A IS INPUT, B IS OUTPUT)
C
      REAL A(12,12),B(12,12)
      DO 1 I=1,N1
      DO 1 J=1,M1
      B(I,J)=A(I,J)
 1    CONTINUE
      RETURN
      END
C**********************************************
      SUBROUTINE EQUATC(A,B,N1,M1)
C**********************************************
C
C     THIS SUBROUTINE EQUATES COMPLEX MATRIX B TO COMPLEX
C     MATRIX A  (A IS INPUT, B IS OUTPUT)
C
      COMPLEX A(12,12),B(12,12)
      DO 1 I=1,N1
      DO 1 J=1,M1
      B(I,J)=A(I,J)
 1    CONTINUE
      RETURN
      END
C*******************************************
      SUBROUTINE MMUL(X,Y,N1,N2,N3,Z)
C*******************************************
```

AD-A144 561   ACTIVE SUPPRESSION OF AEROELASTIC INSTABILITIES ON A   2/2
FORWARD SWEPT WING U..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..   G J PASQUINI
UNCLASSIFIED   01 JUN 84 AFIT/GAE/AA/84J-01   F/G 1/3   NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

```
C
C      THIS SUBROUTINE MULTIPLIES TWO REAL MATRICES :
C
C        X(N1XN2) X Y(N2XN3) = Z(N1XN3)
C
       REAL X(12,12),Y(12,12),Z(12,12)
       DO 3 J=1,N3
       DO 2 I=1,N1
       S=0.
       DO 1 K=1,N2
 1     S=S+X(I,K)*Y(K,J)
 2     Z(I,J)=S
 3     CONTINUE
       RETURN
       END
C*************************************************
       SUBROUTINE MMULC(X,Y,N1,N2,N3,Z)
C*************************************************
C
C      THIS SUBROUTINE MULTIPLIES TWO COMPLEX MATRICES :
C
C        X(N1XN2) X Y(N2XN3) = Z(N1XN3)
C
       COMPLEX X(12,12),Y(12,12),Z(12,12),S
       DO 3 J=1,N3
       DO 2 I=1,N1
       S=CMPLX(0.0,0.0)
       DO 1 K=1,N2
 1     S=S+X(I,K)*Y(K,J)
 2     Z(I,J)=S
 3     CONTINUE
       RETURN
       END
C*******************************************
       SUBROUTINE MTRAN(A,NR,NC,B)
C*******************************************
C
       REAL A(12,12),B(12,12)
C
       DO 1 I=1,NR
       DO 1 J=1,NC
 1     B(J,I)=A(I,J)
C
       RETURN
       END
C*************************************
       SUBROUTINE MTRANC(A,NR,NC,B)
C*************************************
C
       COMPLEX A(12,12),B(12,12)
C
       DO 1 I=1,NR
```

```
      DO 1 J=1,NC
 1    B(J,I)=A(I,J)
C
      RETURN
      END
C***************************************
      SUBROUTINE MADD(A,B,N,M,CONST,C)
C***************************************
C
C     THIS SUBROUTINE ADDS TWO REAL MATRICES :
C
C       A(NXM) + CONST*B(NXM) = C(NXM)
C
      REAL A(12,12),B(12,12),C(12,12)
      DO 1 I=1,N
      DO 1 J=1,M
      C(I,J)=A(I,J)+CONST*B(I,J)
 1    CONTINUE
      END
C****************************************
      SUBROUTINE READ(Y,N,M,JIN)
C****************************************
C
C     THIS SUBROUTINE IS USED TO READ IN A REAL MATRIX
C     Y(NXM) FROM DEVICE NUMBER JIN
C
      REAL Y(12,12)
      NM=N*M
 10   CONTINUE
      READ(JIN,*) I,J,VALUE
      IF(I.EQ.0) RETURN
      Y(I,J)=VALUE
      GO TO 10
      END
C******************************************
      SUBROUTINE WRITE(MAT,N,M,JOUT)
C******************************************
C
C     THIS SUBROUTINE IS USED TO WRITE A REAL
C     MATRIX MAT(NXM) TO DEVICE NUMBER JOUT
C
      REAL MAT(12,12)
      DO 1 L=1,M,6
      K=L+5
      IF(M-L.LT.5) K=M
      DO 2 I=1,N
      WRITE(JOUT,100) (MAT(I,J),J=L,K)
 2    CONTINUE
 100  FORMAT(6E12.5)
      WRITE(JOUT,110)
 110  FORMAT(//)
 1    CONTINUE
```

```
C     WRITE(JOUT,110)
      RETURN
      END
C*****************************************************************
      SUBROUTINE WRITC(MAT,N,M,JOUT)
C*****************************************************************
C
C     THIS SUBROUTINE IS USED TO WRITE A COMPLEX
C     MATRIX MAT(NXM) TO DEVICE NUMBER JOUT)
C
      REAL MATR(12,12),MATI(12,12)
      COMPLEX MAT(12,12)
C
      DO 1 L=1,M,3
      K=L+2
      IF(M-L.LT.2) K=M
      DO 2 I=1,N
      WRITE(JOUT,100) (MAT(I,J),J=L,K)
 2    CONTINUE
 100  FORMAT(6E12.5)
      WRITE(JOUT,110)
 110  FORMAT(//)
 1    CONTINUE
C     WRITE(JOUT,110)
      RETURN
      END
```

VITA

Glenn Justin Pasquini was born on 28 January 1960 in Pittsburgh, Pennsylvania. He attended the University of Pittsburgh from September, 1977 to April, 1981, receiving his Bachelor of Science Degree in Mechanical/Aerospace Engineering. Upon graduating with Honors, he accepted an Aerospace Engineering position with the Stability and Control Branch, Flight Technology Division, Flight Systems Engineering Directorate of the Aeronautical Systems Division. He is currently the Stability and Control/Flight Control analysis Engineer for the B-1B System Program Office. In July, 1981 he entered the part-time Graduate Aeronautical Engineering program at the Air Force Institute of Technology.

Permanent address : 4994 Broughton Place
Dayton, Ohio 45431

AD-A144561

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE DISTRIBUTION UNLIMITED
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GAE/AA/84J-01	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFIT/GAE/AA84J-01
6a. NAME OF PERFORMING ORGANIZATION Air Force Institute of Technology 6b. OFFICE SYMBOL (If applicable) ENY	7a. NAME OF MONITORING ORGANIZATION Air Force Institute of Technology
6c. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, OH 45433
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See item 19	

12. PERSONAL AUTHOR(S)
Glenn Justin Pasquini

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	84 June 1	

Approved for public release: IAW AFR 190-1

Lynn Wolaver 8 Aug 84

LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: Active Suppression of Aeroelastic Instabilities on a Forward Swept Wing Using a Linear Optimal Regulator (Unclassified)

Abstract: Analytical studies were conducted to investigate the potential of applying Optimal Control theory techniques to the synthesis of active flutter suppression control laws. For an example application, a Forward Swept Wing/Fuselage model previously analyzed by Thomas E. Noll of the Air Force flight Dynamics Laboratory was utilized. Through the use of Pade' approximants to represent the unsteady aerodynamic forces, the equations of motion are written in standard state space form. Linear Optimal Regulator theory is then applied to determine particular sets of gains which minimize a quadratic cost function in terms of the states and controls. The control laws are developed at a design flight condition which increases the onset of the lowest instability speed 20% above the

(Continued on reverse)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Robert Calico	22b. TELEPHONE NUMBER (Include Area Code) 513-255-2362 22c. OFFICE SYMBOL AFIT/ENY

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

Block 19 (Continued)

wing bending/torsion instability speed. The optimal control law is then applied at off-design flight conditions to assess the robustness of the Optimal Regulator.

END

FILMED

9-84

DTIC